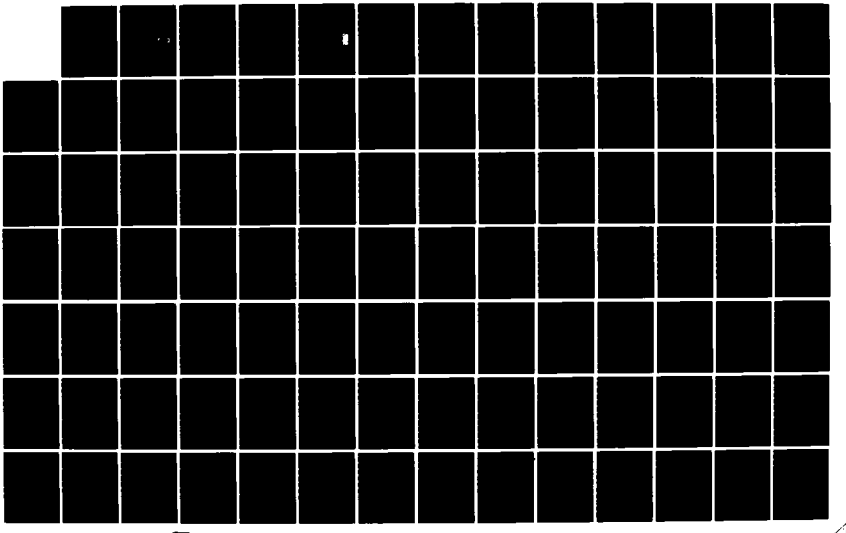
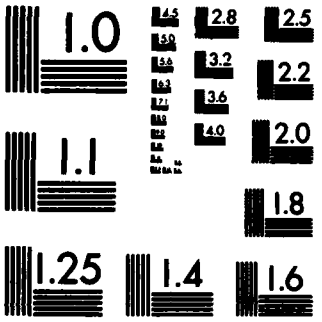
AD-A145 571 AN INVESTIGATION INTO FUZZY CLUSTERING AND CLASSIFICATION(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M R GRAY JUL 84 1/2

UNCLASSIFIED AFIT/CI/NR-84-47T F/G 12/1 NL





UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 84-47T	2. GOVT ACCESSION NO. AD-A145571	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Investigation Into Fuzzy Clustering And Classification		5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael R. Gray		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Univof Missouri-Columbia		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE 1984 13. NUMBER OF PAGES 108
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC ELECTE SEP 17 1984 S D B

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1

5 Sept 84

LYNN E. WOLAVER
Dean for Research and
Professional Development
AFIT, Wright-Patterson AFB OH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

84 09 13 015

AD-A145 571

DTIC FILE COPY

Pattern recognition algorithms based on fuzzy set theory were investigated and compared to their analogs which use traditional, or crisp set theory. The fuzzy K-means clustering algorithm was investigated and the fuzzy K-nearest neighbor and fuzzy 1-nearest prototype classifier algorithms were developed. These pattern recognition algorithms produce membership assignments (values from zero to one) for the samples considered. Thus, a sample's "degree of belonging" in a class can be assessed via these membership assignments.

The fuzzy K-means algorithm serves as an introduction to pattern recognition using fuzzy set theory. By varying a weighting factor(m), used in the fuzzy K-means, over its allowable range ($1 \leq m < \infty$) interesting results were obtained. These results show that the fuzzy K-means algorithm can outperform the crisp version. While execution of the fuzzy K-means algorithm requires more computations than the crisp version the resulting memberships provide more information than the simple cluster assignments produced by the crisp K-means algorithm.

As with the fuzzy clustering algorithm, the membership assignments produced by the fuzzy classification algorithms provides a level of information above that provided by the crisp classifiers. The ideal outcome would be to produce membership assignments which indicate the sample's "degree of belonging" in the class of maximum membership. In attempting to acheive this, a technique for labelled sample membership initialization (used for unknown sample membership assignments) was developed. The method resulted in membership assignments for most unknown samples which were close to one when correctly classified(via maximum membership) and closer to one-half when misclassified.

AFIT/CI/NR 84-47T

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (AU). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: An Investigation Into Fuzzy Clustering And Classification

AUTHOR: Michael R. Gray

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS _____ ( ) b. $_____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME GRADE POSITION

ORGANIZATION LOCATION

STATEMENT(s):

FOLD DOWN ON OUTSIDE - SEAL WITH TAPE

AFIT/NR
WRIGHT-PATTERSON AFB OH 45433

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 73236 WASHINGTON D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

AFIT/ DAA
Wright-Patterson AFB OH 45433

FOLD IN

AN INVESTIGATION INTO FUZZY CLUSTERING AND CLASSIFICATION

A Thesis
Presented to
the Faculty of the Graduate School
University of Missouri-Columbia

In partial fulfillment
of the Requirements for the Degree
Master of Science

---

by

Michael R. Gray

July 1984

Dr. James Keller, Electrical Engineering Dept.

Thesis Advisor

ACKNOWLEDGEMENTS

The author wishes to express his sincere appeciation to Dr. James Keller for the guidance and support he willingly gave during the research and writing of this thesis.

The author also wishes to thank James Givens Jr. for his helpful advice during the research required for this thesis.

The author is especially grateful to his wife, Barbara, who gave her full support and understanding during the months required to complete this thesis.

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

INTRODUCTION

1.1 Introduction

Recognizing patterns, classes, or populations in sample data is an important part of many problems of current interest. Examples such as scene analysis, character recognition, and speech analysis are but a few of the many areas in which pattern recognition techniques have been utilized. Pattern recognition, as a scientific discipline, strives to produce an automated procedure for assigning each element of a set of input data to one of a finite set of classes(1). Thus, a pattern recognition system should reduce the quantity of data present while retaining the information carried. This reduction has become an increasingly important factor as the quantity of data made available by modern digital computer systems continues to grow. Without successful techniques for handling and interpreting data, the sheer quantity produced can become a burden rather than an aid. Pattern recognition techniques

are recognized as providing useful approaches to solution of this problem(1). As a result, such techniques have been used extensively in the design of computerized information processing systems(1).

The pattern recognition or classification model is composed of the following three components: a transducer, a feature extractor, and a classifier(2). The transducer senses the input and converts it into a form suitable for machine processing. The feature extractor receives the output of the transducer and extracts a set of feature measurements which represent the nature of the data. Finally, these feature measurements are received by a classifier which assigns the input data elements to one of the possible classes.

Each of the components described above is dependent to a varying degree on the particular problem being considered. The design or specification of a suitable transducer is highly problem dependent and is not considered in this report. In general, of the remaining two components, the problem of feature extraction is much more problem dependent than that of classification. Many useful techniques for feature extraction exist, some of which are discussed in (1,2). While the problem of feature extraction is not considered specifically in this study, it is important to realize the connection between it and the classification

problem. The better the input is represented by the feature extractor, the easier the classification task becomes(2).

1.2 Approaches to Classifier Design

The approaches used to design automatic pattern recognition systems can be divided into three categories(1). These categories are template matching, decision theoretic methods, and syntactic recognition. Template matching is based on the idea of comparison of input samples to a set of stored templates which represent each of the possible classes. The decision theoretic techniques attempt to formulate a set of classification rules which are defined by a function of the sample features. The third technique, syntactic pattern recognition, suggests that the sample patterns can be represented using a hierarchical structure present in the data. Each of these methods, while utilizing different procedures, results in some form of decision rule for data classification.

The template matching approach is based on a comparison technique. That is, an unknown sample is compared to a set

of templates stored in the system until a match is obtained. An application where templace matching has been utilized successfully is that of character recognition(1). By limiting the form of the characters under consideration, such as typed characters of uniform size, the problem can be reduced to a manageable form. Typically, a set of measurements which allow unique representation of each character allowed are available to the system. Then, an input sample(character) is examined and the same set of measurements are recorded. As a result, all that is necessary to classify the sample is to compare the measurements obtained to those stored in the system until a match is found. Clearly, this technique places exact restrictions on the samples under consideration(1). Also, if the number of characters allowed is very large the storage requirements may be burdensome and the time required to search for a match will be excessive.

The decision theoretic approach may be subdivided into either deterministic or probabilistic techniques. The deterministic techniques utilize analytic functions to provide a functional description of the decision rule(2). As an example of a deterministic pattern classifier, consider the 1-nearest neighbor pattern classifier. The nearest neighbor classifier finds the nearest labeled sample(i.e. of known class) using a distance measure. The distance measure can be of varying type; the Euclidian

distance is one commonly used. Once the nearest neighbor has been found, the sample pattern is assigned to the class of this closest neighbor. Ties are resolved arbitrarily.

The probabilistic mathematical techniques utilize the statistical properties of the pattern classes to achieve a decision rule(2). A probability density function describing the distribution of the class is obtained and used to formulate the decision rule. One example of this method is the Bayes classifier. The Bayes classifier is typically used when the density functions are assumed to be multivariate normal (i.e. the data is normally distributed)(1). The mean and covariance matrix corresponding to the classes under consideration are obtained by either direct calculation or an approximation technique. With these parameters, the normal density function is completely defined. The density function for each class is then evaluated for the sample pattern under consideration, these values are combined with the probability of occurrence of each class, and the sample is then assigned to the class for which the resultant value is a maximum. As with the nearest neighbor classifier, ties are resolved arbitrarily. The mathematical techniques are generally not as restrictive as those of the template matching method. Nevertheless, these techniques are also dependent upon the application considered.

The syntactic approach to the pattern recognition problem utilizes the structure existing in the sample classes. Formal language theory is applied to describe the levels of structure present in terms of a particular grammar. Of course, this approach presupposes the existence of some form of recognizable structure. As a result, syntactic techniques are best applied to problems in which the structure present can be characterized in some concise form. Syntatic techniques have been used in pictorial pattern recognition as well as in other areas. Of the three approaches discussed, the syntactic approach is the least developed. But, its use has been receiving increased attention recently. The theory of syntactic pattern recognition is covered extensively in (3), and (1,2) provide an introductory look at the topic.

The work presented here considers mathematical methods for classifier design. More specifically, an investigation into algorithms based on fuzzy set theory is presented with comparisons to their crisp analogs. The algorithmic pattern recognition techniques discussed are deterministic in nature, except for one probabilistic method based on Bayes decision theory. Of course, since the lines which separate the three approaches are not hard and fast, it is important to be able to draw from any of the three when developing an effective classifier.

1.3 Introduction to Fuzzy Sets

The theory of fuzzy sets was developed by Lofti Zadeh in 1965(4). The impetus behind the introduction of the fuzzy set was to provide a means of defining categories which are inherently imprecise(5). While it is a relatively new concept, the theory is a natural extension of traditional set theory. Since the introduction of fuzzy set theory, the terms "hard" and "crisp" have been used to describe sets conforming to the traditional set theory. Although it has taken some time for its use to spread, the theory of fuzzy sets has been applied successfully to a variety of areas. These include medical diagnosis, linguistic modelling, artificial intelligence, and scene analysis as well as pattern recognition. The results achieved in these applications are useful and have stimulated further research in the area.

Prior to the introduction of fuzzy sets, probability theory was the primary mathematical means of describing imprecision. Although many people still believe that probability theory is all that is needed to handle problems which are inherently imprecise, failure to examine all possible methods of achieving a solution will very likely

lead to a less than optimal solution. Upon comparison of the imprecision, or fuzziness which is modelled by fuzzy set theory to the randomness which probability theory models so well, it should be clear that the two theories are distinct. Consider the statement: "You are nearly correct". Using probability theory this would be modelled as: "There is an X% possibility that you are correct."(X could be something near 90). But the intent of such a statement is to say that the response you supplied is close to the correct one, not, as probability theory suggests, that there is a good chance you are correct. Alternatively, fuzzy set theory models the statement as: "The correctness, on a scale from zero to one, of your answer is X."(X could be near 0.9). Now this is the true intent of the statement given above. Thus, the difference between fuzzy modelling and stochastic modelling is that fuzzy set theory handles imprecision easily whereas probability theory is best suited to random processes(5). So, it is not a matter of which theory is best, but instead which theory is best suited to the problem at hand.

The basis of fuzzy set theory is that set elements may take on a membership other than complete membership(membership=1) or non-membership(membership=0). Thus, as is often the case in real world situations, a set may consist of elements with varying degrees of similarity. The measure of similarity is assigned via a membership function. In traditional, or "crisp" set theory the

membership function values are restricted to zero or one. But, in fuzzy set theory an element's membership function may take on any value in the closed interval [0,1] . Thus more flexibility results by using fuzzy set theory to describe classes and their members. For a more complete discussion of the theory of fuzzy sets, the reader is referred to (6) which provides an excellent and thorough presentation of the theory.

The following examples illustrate the usefulness of fuzzy sets for describing classes which exist in the world. Consider the class of all young people in the world. Clearly, we must define what attribute a person must possess to be considered young before determining who belongs in the class. One might say that a person is young if they are less than some particular age. Then, using crisp set theory to describe this class we simply say that everyone less than the given age is young and all others are not. On the other hand, using fuzzy set theory we assign a membership in the class of young people to all persons considered. Thus, using age to define young people, a five year old person might have a membership of 0.95 in the class while a ninety year old person might have a membership of 0.1 in the class. Clearly, the latter description provides more information to the observer and as a result should be more useful to someone concerned with young people of the world. As another illustration, consider the classic example of the

set of all bald persons. As before we must define what attribute a person must posses to be considered a member of the set. Should a person whose hairline has receded a couple of inches be a member? Of course a person with no hair will be a member. But where do we set the defining line for baldness? Without going any further with this example it should be clear that crisp set theory will not provide much help in identifying the set of bald people unless we all can agree at what stage a person is considered bald.

With just the two examples presented above it should be clear that there are many cases in the world where the models based on crisp set theory fall short of providing a useful description of things, people, or places. So, as Professor Zadeh proposed, the use of fuzzy set theory may indeed perform better in these cases.

CHAPTER 2

FUZZY CLUSTERING

2.1 Introduction

As discussed in chapter one, evaluation of large data sets can be a difficult task simply because of the volume of data present. One way of reducing the data is to use a clustering procedure to extract information from the raw data. Roughly speaking, clustering procedures yield a data description in terms of clusters, or groups of data points which possess some form of similarity(2).

When the clustering procedures are based on crisp set theory a sample in the data set must be classified as belonging to one and only one cluster(1). This constraint is imposed by the mathematical model based in crisp set theory. As an example, consider the case of a set of data samples taken from three classes, one being a hybrid of the other two classes resembling each non-hybrid to the same degree. If we have no prior knowledge of the actual number

of classes present and partition the data into two clusters, the following should occur. If the two non-hybrid classes are separable, samples taken from these classes will be placed in different clusters. But, what of the samples from the hybrid class? These samples will likely be divided into the two classes resulting in clusters which are distorted from their natural shape and density. In addition, the samples from the hybrid data will be lost amongst the other samples with nothing to indicate a difference in their origin.

Alternatively, a fuzzy clustering technique does not have the same constraint as that imposed by crisp set theory. Instead, samples are assigned membership in all classes(5). Returning to the example problem described above, the two non-hybrid classes will have high membership(close to one) in one cluster and low membership(close to zero) in the second cluster. Of course, each non-hybrid will have high membership in different clusters. Now, consider the samples from the hybrid class. Since they do not resemble one non-hybrid more than another they will be assigned membership in each cluster very close to one-half. Thus, these samples will be recognized as not belonging to one cluster more than another, as they should be. This example points out the essence of fuzzy clustering. That is, fuzzy clustering procedures do not force a sample into one and only one cluster. Instead, a sample's "degree of belonging"

in a particular cluster can be interpreted via its membership assignments.

In both cases of the example given above, as in most clustering procedures, the technique in both crisp and fuzzy methods is to assign individual data points to a cluster such that the resulting clusters produce a natural grouping of the data. Of course, we must define what is meant by a natural grouping. Typically this is defined by a measure of similarity between samples as well as a criterion for evaluating the partition which results from the clustering procedure(2). Thus, the choice of similarity measure and criterion function in a clustering procedure strongly influences the type of clusters obtained.

2.2 Similarity Measures

The similarity measure used in a clustering procedure defines what mathematical properties of the data should be used to identify clusters(5). Properties such as distance, angle, curvature, symmetry, and intensity are some which may be of interest. Clearly, no one measure of similarity will

be universally applicable. Often the choice of one measure over another is a subjective one, with considerations of prior knowledge and ease of implemetation playing a role.

The most obvious measure of similarity between two samples is the distance between them(2). Of course there are several ways in which the distance between two points can be defined. The Euclidian distance squared between two sample vectors X_j and Z_j

$$D^2 = (X_j - Z_j)^t (X_j - Z_j)$$

is one commonly used measure of similarity, with a smaller distance corresponding to a greater similarity. Use of the Euclidian distance to test similarity in a clustering procedure produces clusters which are hyperspherical(2). The Malahanobis distance from a sample vector X_j to a mean vector M ,

$$D^2 = (X_j - M)^t \; C^{-1} \; (X_j - M)$$

is a useful measure of similarity when the statistical properties of the data are being considered(here C^{-1} is

the inverse covariance matrix of the sample data). This similarity measure produces clusters which are hyperellipsoids(2). Distance measures as a form of comparison are by no means the only useful similarity measures.

A nonmetric similarity measure between two vectors X and Z,

$$S = \frac{X^t Z}{\|X\| \|Z\|}$$

represents the cosine of the angle between the two vectors X and Z. This similarity measure is a maximum when the vectors are oriented in the same direction with respect to the origin. Thus, this measure is useful when clusters tend to align themselves along the principle axis(1).

The similarity measures given above are some of those commonly used in clustering problems. Of course many more similarity measures exist, some of which are discussed in (1,2,5). For the pattern recognition algorithms considered in this report, the Euclidian distance measure is used. This similarity measure was chosen since on the whole very little prior knowledge concerning the types of clusters to

expect in the test data was available. In addition, by using this measure exclusively, variability in results due solely to the use of different distance measures was eliminated.

2.3 Criterion Functions

In order to obtain the set of K clusters, or subsets of a sample set which are the "most desirable", we need to define a criterion function which measures the quality of the clusters found. The "most desirable" clusters are those which contain samples which are somehow more similar than samples contained in a different cluster(2). Thus, once such a criterion function is defined, partitioning the data such that the criterion function is an extreme(maxima or minima) will produce the "most desirable" clusters obtainable under the given criteria. Of course, the result does not necessarily represent the naturally occurring clusters, if any, in the set of samples. The extent to which the clusters obtained represent the naturally occurring clusters is dependent upon the particular choices for a similarity measure and criterion function(2).

The most widely used clustering criterion function is the sum-of-squared-error criterion(1). Let n_j be the number of samples in a proposed cluster and let m_j be the mean of those samples,

$$m_j = \frac{1}{n_j} \sum_{j=1}^{n_j} x_j$$

The sum of squared errors is defined as,

$$J_e = \sum_{j=0}^{k} \sum_{i=0}^{n_j} \|x_j - m_j\|^2$$

where K is the number of clusters and n_j is the number of vectors in the jth cluster. The interpretation of this criterion function is as follows. For a given cluster, the mean vector m_j is the best representative of the samples in the cluster in the sense that it minimizes the sum of squared lengths of the "error" vector $\|x_j - m_j\|$ (2). As a result, J_e measures the total squared error incurred by representing the n samples by the K cluster centers. Then the optimal partition as defined by this criterion function is one which minimizes J_e . Clusters resulting from the use of the sum-of-squared-error as the criterion function are often called "minimum variance" partitions(2). The fuzzy

analog to the sum-of-squared-error criterion is very much like the one given above. The difference is that the distance measure for each vector x is multiplied by x's membership in the class raised to the power m, where m is a weighting factor usually taken as two.

The clustering problems best suited to the use of J_e are those which form well separated compact "clouds". One problem arising from the use of this criterion when there is a large difference in the number of samples in different clusters is that the large clusters may be split because of the small reduction in squared error being multiplied by many times in the sum(5). Situations producing such a problem often occur when there exists single points well away from the more dense regions of the cluster.

There are other useful criterion functions, several of which are discussed in (2). The common feature of the criterion function presented above as well as those in (2) is that they model the clustering problem as one in which the samples form well separated "clouds" of points. While this model may be reasonable in some cases it does not represent the majority of the clustering problems which are of concern. As a simple example consider the case of the "cloud within a cloud", a dense cluster embedded in the center of a diffuse cluster. Clearly, utilizing a criterion which uses the model described above will not likely produce

a useful partition. Nevertheless, criterion such as the minimum squared error function are often used as a starting point, then a different criterion function must be devised if the results are not meaningful.

2.4 Clustering Methods

2.4.1 Hierarchical Methods

This group of methods were originally utilized in the field of biological taxonomy where individuals are grouped into species, species into genera, genera into families, and so on(2). Hierarchical clustering contains both agglomerative(merging) and divisive(splitting) techniques. In both cases the procedure is to form new clusters by reallocating membership of one point at a time, based on a given similarity measure(5). Thus, the resulting clusters form a hierarchy of nested clusters. Because of their conceptual and computational simplicity, hierarchical methods are among the best known(2). They are suitable for use when the underlying structure of the data is dendritic(5). An introductory look at the methods of

hierarchical clustering is presented in (2). In addition, a discussion of fuzzy hierarchical clustering techniques is presented in (7).

2.4.2 Graph-Theoretic Methods

In this group the set of samples is regarded as a node set, and edge weights between pairs of nodes can be based on a similarity measure between pairs of nodes(5). The clustering criterion may be some measure of connectivity between groups of nodes. Breaking of edges in a minimal spanning tree to form subgraphs is an often used graph-theoretic clustering strategy(5). The benefit of graph theoretic techniques is that they allow consideration of more intricate structures than the isolated "cloud-like" clusters produced by the mathematics of normal mixtures and minimum-variance partitions(2).

2.4.3 Objective Function Methods

These methods generally allow the most precise(but not necessarily more valid) formulation of the clustering criterion(5). Objective function methods make use of criterion functions, such as those described above, as a measure of each clustering candidates "desirability". Thus, the optimum clusters under these methods are those which produce local extrema of the objective function(5). The K-means algorithm described in the following section is of this type.

2.5 K-means Algorithm

This algorithm, in both the fuzzy and crisp versions is based on the minimization of the within-group sum of squared error criterion. Both the fuzzy and crisp algorithms are given below. The crisp K-means algorithm is included to provide a comparison between fuzzy and crisp clustering results. The notation used in the algorithms is as follows.

K ≡ number of clusters specified

n ≡ number of data samples

$n_i \equiv$ number of sample vectors in the ith cluster

$\{X\} \equiv$ the set of n sample vectors

$m \equiv$ weighting factor

$U^l \equiv$ membership function array for the lth iteration

$u_{ij} \equiv$ the membership of the jth vector in the ith cluster

$\{V^l\} \equiv$ the set of K fuzzy cluster centers for the lth iteration

$\{Z^l\} \equiv$ the set of crisp cluster means for the lth iteration

$\|A\| \equiv$ any matrix norm(for example the max of the absolute values of all elements)

The procedures for the fuzzy K-means algorithm are,

```
BEGIN
  Set K, 2≤K<n
  Set ε , ε≥0
  Set m, 1≤m<∞
  Initialize U⁰
  Initialize l=0
  DO UNTIL( ‖U^l-U^(l-1)‖ < ε )
    Increment l
    Calculate {V_i^l} using 2.5a and U^(l-1)
    Compute U^l using 2.5b and {V_i^l}
  END DO UNTIL
END
```

$$2.5a \quad V_i = \frac{\sum_{j=1}^{n} (u_{ij})^m x_j}{\sum_{j=1}^{n} (u_{ij})^m}$$

$$2.5b \quad u_{ij} = \frac{1/\|x_j - v_i\|^{2/(m-1)}}{}$$

$$\sum_{k=1}^{K} (1/\|x_j - v_k\|)^{2/(m-1)}$$

The crisp K-means algorithm is as follows.

```
BEGIN
  Set K, 2≤K≤n
  Initialize {Z_i^0} by arbitrary assignment
   as vectors in the sample set
  Initialize l=0
  DO UNTIL ( ||Z_i^l - Z_i^(l-1)|| < ε, ∀ i = 1,2,...,K )
    Increment l
    Assign each x∈{X} to the jth cluster if
     ||x - Z_j^(l-1)|| < ||x - Z_i^(l-1)|| ∀ i=1,2,...,K
    Compute {Z_i^l} using 2.5c
  END DO UNTIL
END
```

$$2.5c \quad Z_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j$$

As the statements of the algorithms illustrate, both are relatively simple procedures. Although neither of these algorithms have a general convergence proof associated with them, they both have been shown to provide useful results(1,5). In the case of the fuzzy K-means algorithm, a proof of convergence under certain conditions to a local minimum of the within-group sum of squared error criterion exists(5).

While both algorithms are useful in determining the existence of a set of K clusters in some data sets, the correct choice of K is no straightforward task. Often the data must be run for several values of K. Then the results of all runs must be interpreted (usually by hand) to determine the number of naturally occurring clusters, if any. One advantage of the fuzzy K-means algorithm is that the interpretation is eased by the availability of the membership function array. When the memberships show most samples with a high membership in only one cluster then this suggests the choice for K which produced the results may best represent the number of naturally occurring clusters.

The initialization steps required for these algorithms are quite different. For the crisp K-means algorithm a set of K initial cluster centers must be chosen. Usually a random assignment will produce good results. Alternatively, the fuzzy K-means algorithm requires a little more effort in order to obtain useful results. The initial membership function array can not in general be assigned arbitrarily. One procedure for initialization of the array is to obtain a *crisp partition and* then "fuzzify" it by changing each vectors memberships so that they share membership among the classes with their maximum membership in the class which the crisp partition placed them(5).

A great deal of research concerning the fuzzy K-means algorithm has been conducted. Several individual as well as joint efforts have been completed by James Bezdek and Joseph Dunn(5,8,9,10). On the whole, as indicated by their research results, the fuzzy K-means algorithm is a useful tool for cluster analysis. Additional results and a comparison between the fuzzy and crisp algorithms are given in chapter four.

FUZZY CLASSIFIERS

3.1 Introduction

While clustering procedures are utilized when the nature of a set of unlabelled samples is being investigated, classification routines have a different purpose. Given a set of unlabelled samples, a classification algorithm should be able to determine their correct classification. There are several approaches to the classifier problem, as discussed in chapter one. In this chapter the nearest neighbor and nearest prototype classifiers are considered.

Both the nearest neighbor and nearest prototype classifiers utilize labelled samples and a distance measure to determine classification. In the case of the nearest neighbor classifier the labelled samples are used directly. The nearest prototype classifier compares the samples of unknown class to a set of prototypical samples representing the possible classes.

3.2 Nearest Neighbor Classifiers

The nearest neighbor classifiers require no preprocessing of the labelled sample set prior to their use. The crisp nearest neighbor classification rule assigns an input sample vector y, of unknown classification, to the class of its nearest neighbor(1). This idea can be extended to K nearest neighbors with the vector y being assigned to the class which is represented by a majority amongst the K nearest neighbors. Of course, when more than one neighbor is considered the possibility that there will be a tie among classes which have a maximum number of neighbors in the group of K nearest neighbors exists. One simple way of handling this problem is to restrict the possible values of K. For example, given a two class problem, if we restrict K to odd values only no tie will be possible. Of course, when more than two classes are possible this technique is not useful. The means of handling the occurrence of a tie is as follows. The sample vector is assigned to the class, of those classes which tied, for which the sum of distances from the sample to each neighbor in the class is a minimum. Of course, this could still lead to a tie, in which case the assignment is to the last class encountered amongst those which tied, an arbitrary assignment. Clearly, there will be cases where a vector's classification becomes an arbitrary assignment no matter what additional procedures are included in the algorithm.

3.2.1 Crisp Nearest Neighbor Algorithm

Let $W = \{x_1, x_2, \ldots, x_n\}$ be a set of n labelled samples. The algorithm is as follows.

```
BEGIN
  Input y, of unknown classification
  Set K, 1≤K≤n
  Initialize i=1
  DO UNTIL( K nearest neighbors found)
    Compute distance from y to xi
    IF (i ≤ K ) THEN
      Include xi in the set of K nearest neighbors
    ELSE IF ( xi is closer to y than
              any previous nearest neighbor ) THEN
      Delete farthest in the set of K nearest neighbors
      Include xi in the set of K nearest neighbors
    END IF
    Increment i
  END DO UNTIL
  Determine the majority class represented in the set
   of K nearest neighbors
  IF (a tie exists ) THEN
    Compute sum of distances of neighbors in each
      class which tied
    IF ( no tie occurs ) THEN
      Classify y in the class of minimum sum
    ELSE
      Classify y in the class of last minimum found
    END IF
  ELSE
    Classify y in the majority class
  END IF
END
```

3.2.2 Fuzzy Nearest Neighbor Classifier

While the fuzzy K-nearest neighbor procedure is also a classification algorithm the form of its results differ from the crisp version. The fuzzy K-nearest neighbor algorithm assigns class membership to a sample vector rather than assigning the vector to a particular class. The advantage is that no arbitrary assignments are made by the algorithm. In addition, the vector's membership values should provide a level of assurance to accompany the resultant classification. For example, if a vector is assigned 0.9 membership in one class and 0.05 membership in two other classes we can be reasonably sure the class of 0.9 membership is the class to which the vector belongs. On the other hand, if a vector is assigned 0.55 membership in class one, 0.44 membership in class two, and 0.01 membership in class three then we should be hesitant to assign the vector based on these results, although we can feel confident that it does not belong to class three. In such a case the vector might be examined further to determine its classification. Clearly the membership assignments produced by the algorithm can be useful in the classification process.

The basis of the algorithm is to assign membership as a function of the vector's distance from its K nearest neighbors and those neighbors membership in the possible classes. The fuzzy algorithm is similar to the crisp version in the sense that it must also search the labelled sample set for the K nearest neighbors. Beyond obtaining these K samples, the procedures differ considerably.

3.2.2.1 Fuzzy Nearest Neighbor Algorithm

Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of n labelled samples. Also let $u_i(x)$ be the assigned membership of the vector x(to be computed),and u_{ij} be the membership in the ith class of the jth vector of the labelled sample set.

```
BEGIN
  Input x, of unknown classification
  Set K, 1≤1≤n
  Initialize i=1
  DO UNTIL ( K nearest neighbors to x found )
    Compute distance from x to xi
    IF (i ≤ K ) THEN
      Include xi in the set of K nearest neighbors
    ELSE IF ( xi closer to x than
                any previous nearest neighbor ) THEN
      Delete the farthest of the K nearest neighbors
```

```
      Include x_j in the set of K nearest neighbors
    END IF
  END DO UNTIL
  Initialize i=1
  DO UNTIL ( x assigned membership in all classes )
    Compute u_i (x) using 3.2.2.1a below
    Increment i
  END DO UNTIL
END
```

$$3.2.2.1a \quad u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}\left(1/\|x-x_j\|^{2/(m-1)}\right)}{\sum_{j=1}^{K}\left(1/\|x-x_j\|^{2/(m-1)}\right)}$$

3.2.2.2 Physical Interpretation

The interpretation of the algorithm is given in terms of the following example. As stated previously, selection of the K nearest neighbors from the labelled sample set is straightforward. So, with K=3 the example proceeds assuming the 3 nearest neighbors of x are x_1, x_2, and x_3. The class memberships for these three sample vectors are given as:

u_{ij} = membership of jth sample in the ith class, j=1,2,3.

The distances of x from x_1, x_2, and x_3 are d_1, d_2, and d_3 respectively.

Now according to 3.2.2.1a,

$$u_i(x) = \frac{u_{i1}(1/d_1)^{2/m-1} + u_{i2}(1/d_2)^{2/m-1} + u_{i3}(1/d_3)^{2/m-1}}{\sum_{j=1}^{3}(1/d_j)^{2/m-1}}$$

Thus, the assigned memberships of x are influenced by the inverse of the distances from the nearest neighbors and their class memberships. The inverse distance serves to weight a vector's membership more if it is closer and less if farther from the vector under consideration. The labelled samples can be assigned class memberships in one of two ways. First, they can be given complete membership in their known class and non-membership in all other classes. The second alternative is to assign the samples membership based on distance from their class mean or distance from labelled samples of the other class or classes, and then use the resulting memberships in the classifier. Both of these

techniques have been used in this study and the results are reported in chapter four.

3.3 Nearest Prototype Classifiers

These classifiers bear a marked resemblance to the 1-nearest neighbor classifier. Actually, the only difference is that for the nearest prototype classifier the labelled samples are a set of class prototypes whereas in the nearest neighbor classifier we use a set of labelled samples which are not necessarily prototypical. Of course, the nearest prototype classifier could be extended to multiple prototypes representing each class, similar to the K-nearest neighbor routine. Nevertheless this study considers only the 1-nearest prototype classifier in both a crisp and fuzzy version. The prototypes used for these routines are taken as the class means of the labelled sample set.

3.3.1 Crisp Nearest Prototype Algorithm

Let $W = \{Z_1, Z_2, \ldots, Z_k\}$ be the set of K prototype vectors representing the K classes.

```
BEGIN
  Input x, vector to be classified
  Initialize i=1
  DO UNTIL (distance from each prototype to x computed)
    Compute distance from Z_i to x
    Increment i
  END DO UNTIL
  Determine minimum distance to any class prototype
  IF ( tie exists ) THEN
    Classify x as last class found of minimum distance
  ELSE
    Classify x as class of closest prototype
  END IF
END
```

3.3.2 Fuzzy Nearest Prototype Algorithm

As above, let $W = \{Z_1, Z_2, \ldots, Z_k\}$ be the set of K prototypes representing the K classes.

```
BEGIN
  Input x, vector to be classified
  Initialize i=1
  DO UNTIL (distance from each prototype to x computed)
    Compute distance from Z_i to x
    Increment i
```

```
  END DO UNTIL
  Initialize i=1
  DO UNTIL (X assigned membership in all classes)
    Compute u_i(x) using 3.3.2a below
    Increment i
  END DO UNTIL
END
```

$$3.3.2a \quad u_i(x) = \frac{1/\|x-x_i\|^{2/(m-1)}}{\sum_{j=1}^{K}(1/\|x-x_j\|^{2/(m-1)})}$$

The difference between 3.3.2a and 3.2.2.1a is that membership in each class is assigned based only on the distance from the class prototype. This is because the prototypes should naturally be assigned complete membership in the class which they represent.

CHAPTER 4

RESULTS AND CONCLUSIONS

4.1 Introduction

The results presented in this chapter were produced by software implementation of the algorithms presented in chapters three and four. The software was developed using Fortran 77 on a Perkin-Elmer 3220. In addition, UMC Core Graphics support software was utilized to allow a geometric interpretation of the two-dimensional clustering and classification results.

4.2 Test Data

Four labelled data sets were utilized to test the algorithms. The data sets and their attributes are as follows.

Data Set name	Number of classes	Number of vectors	Number of features per vector
IRIS	3	150	4
IRIS23	2	100	4
TWOCLASS	2	242	4
LANDSAT	4	32018	4

The IRIS data is that of Anderson(11). This particular data set has been utilized extensively by researchers in the area of cluster analysis since 1936, when R.A. Fisher first used it to illustrate the concept of linear discriminant analysis(5). The data represents three subspecies of irises, with the four feature measurements being sepal length, sepal width, petal length, and petal width, all in centimeters. There are fifty vectors per class in this data set. The IRIS23 data set is a subset of the IRIS data. It

includes classes two and three, the non-separable classes, of the IRIS data.

The TWOCLASS data set is an artificially generated normally distributed set of vectors. This data set was included because classification results from a Bayes classifier were available to use in the comparison. This data set contains 121 samples per class.

The third data set is a set of images taken by Landsat-2 on April 22, 1981 and August 9, 1981. Features one and two were produced in April and features three and four in August. Features one and three were produced by identical sensor types as were two and four. This data set was used exclusively in the clustering evaluation. As with the TWOCLASS data, results from a different clustering procedure ran on this data was available for comparison. For additional information concerning the data source refer to (12). The clustering results available were produced by a statistically oriented algorithm entitled SEARCH which is also described in (12).

The IRIS data and TWOCLASS data sets were utilized in evaluation of both the clustering and classification algorithms.

4.3 Clustering Results and Computational Requirements

4.3.1 Clustering Results

As a basis for comparison the results of the fuzzy clustering algorithm are reported as a crisp partition wherein a vector is assigned to the cluster of maximum membership. With these results, shown in Tables 1, 1a, and 1b. a comparison of the crisp and fuzzy algorithms can be made. The percentages given in the tables indicate the rate of correct classification, for individual classes and combined results.

The results are presented in the form of confusion matrices. These matrices are organized as follows. The count of samples listed in each row are those which belong to the corresponding class and the count of samples listed in each column are those placed into the corresponding cluster. Thus, the rows give the vectors in the corresponding class and the columns give the resultant cluster assignments.

Consider first the results shown for the IRIS and TWOCLASS data sets. In the case of the TWOCLASS data the results are the same for both crisp and fuzzy clustering. The results of the two clustering procedures do show a difference for the IRIS data, although the difference in error rate is less than 1%, hardly significant.

Next, examine the results shown for the LANDSAT data. First of all, the numbers in the results of the SEARCH procedure differ by a scale factor because they are reported in terms of acreage whereas the other results are reported in terms of pixel count. Comparing only the results of the fuzzy clustering to the crisp clustering it should be clear that the fuzzy K-means algorithm performed much better than the crisp version. Actually, the only reason the crisp algorithm's results show an overall rate of correct labelling above fifty percent is because the majority of the sample points are from a single class.

Now, if we compare the results of the fuzzy K-means algorithm to those of the SEARCH algorithm the following observations can be made. First of all the overall rate of correct labelling for SEARCH is higher than that of the fuzzy results. But, by examining the results of the individual classes we can see that the fuzzy clustering routine did better, on the average, for the individual classes. In addition, while the SEARCH procedure is

considered unsupervised clustering, it does involve user interpretation of intermediate results(a much larger number of clusters), which is then given to the algorithm in terms of desired cluster combinations so that the final cluster count will be as specified, which in this case is four(12).

From the above results it should be clear that the fuzzy K-means performs as well as, and in some cases better than the crisp K-means algorithm.

4.3.2 Role of the Weighting Factor

The weighting factor (m) used in the fuzzy algorithms influences the results of these algorithms in an interesting manner. The results of the K-means algorithm when m is varied over a range of values are presented in Tables 2 and 2a.

The first thing to notice from the results listed in Table 2 is that the rate of correct classification increases without exception for both data sets as the value of m is increased over the range. In addition, as m is increased

the resulting memberships become "fuzzier", as expected(5). That is, on the average, the membership assignments given are closer to 0.5, the region where it would seem the membership of a vector would be more difficult to distinguish. Nevertheless, in the empirical results presented in Tables 2 and 2a, the "fuzzier" memberships do not cause the error rate to increase, instead it decreases. While these results are not conclusive, they do show that the fuzzy K-means algorithm can outperform the crisp K-means algorithm.

4.3.3 Computational Requirements

The computational requirements of the crisp and fuzzy K-means algorith will now be considered. The number of multiplications and additions are compared in the general case and for a particular example. The count of multiplications and additions for each algorithm are reported in terms of the parameters listed below.

K ≡ number of cluster specified

N ≡ number of data samples

n ≡ number of features per sample

	Fuzzy	Crisp
Multiplications	KN(3+2n)	Kn(2+N)
Additions	3KN(1+n)	2Kn(1+N)

Using the parameter for the IRIS data set, the following particular example is given.

K = 3

N = 150

n = 4

	Fuzzy	Crisp
Multiplications	4950	1824
Additions	6750	3624

Without a doubt there is a trade-off involved when using the fuzzy K-means algorithm as opposed to the crisp K-means

algorithm. But, the fuzzy K-means algorithm provides more information, in the form of cluster memberships, than the crisp K-means algorithm.

4.4 Classifier Results and Computational Requirements

As with fuzzy clustering, results of the fuzzy classifications are reported in terms of a crisp partition wherein a sample vector is assigned to the class of maximum membership. The classifications are obtained using the "leave one out" technique. The procedure is to leave one sample out of the data set and classify it using the remaining samples as the labelled data set. This technique is repeated until all samples in the data set have been classified. In addition, in order to evaluate one technique used to initialize memberships of the labelled samples used in the classifier the IRIS23 data set was created by using only class two and three of the IRIS data set. This was necessary because the initialization technique will only work on two class classification problems.

4.4.1 Results of Nearest Neighbor Classifiers

Before comparing the results produced by the nearest neighbor algorithms, the types of labelling techniques used for the fuzzy classifier are explained. Three different techniques of membership assignment for the labelled data are considered. The first method, a crisp labelling, is to assign each labelled sample complete membership in its known class and zero membership in all other classes. The second technique utilized assigns membership based on the procedure presented in (13). This technique works only on two class data sets. The procedure assigns a sample membership in its known class based on its distance from the mean of the labelled sample class. These memberships range from one to one-half with an exponential rate of change between these limits. The sample's membership in the other class is assigned such that the sum of the memberships of the vector equals one. A more detailed explanation of this technique is given in (13). The third method considered assigns memberships to the labelled samples according to a K-nearest rule. The K(not K of the classifier) nearest neighbors to each sample (x) are found and then membership in the known class i is assigned according to the following equation.

$$u_i(x) = 0.51 + (n_i/K)*0.49$$

Membership assignments in the remaining classes are according to(C ≡ number of classes),

$$u_j(x) = (n_j/K)*0.49 \quad j = 1, 2, \ldots, C \quad j \neq i$$

The value n_i is the number of the neighbors found which belong to the ith class and the value n_j is the number of the neighbors found which belong to the jth class. This method attempts to "fuzzify" the memberships of the labelled samples which are in the class regions which intersect in the sample space and leave the samples which are well away from this area with complete membership in the known class. As a result, an unknown sample lying in this intersecting region will be influenced to a lesser extent by the labelled samples which are in the "fuzzy" area of the class boundary. This initialization technique would work better on the problem of the "cloud within a cloud" discussed in section 2.3.

Thus, with these three initialization techniques three sets of results of the fuzzy K-nearest neighbor classifier are produced.

These results are presented in Tables 3 and 4. Upon comparison of the results of the crisp classifier and the fuzzy classifier with crisp initialization we can see that on the average these procedures have equal error rates. In addition, the fuzzy classifier which uses the second initialization technique described produced nearly equal results. Although not reported in the tables, the results of this fuzzy classifier using the membership assignment rule described in (13) did not produce memberships for the misclassified vectors which suggest they actually belong do a different class. Instead this second initialization technique causes an overall reduction in the values of memberships assigned with most of the samples given majority memberships less than 0.7. But the nearest neighbor initialization technique does seem to produce membership assignments which give an indication of degree of correctness of classification.

Examining the results given in Table 4 for the K-nearest neighbor classifier with nearest neighbor sample membership initialization, the following observations can be made. First of all, the results show a somewhat lower overall error rate. But, more importantly, the number of misclassified vectors with high assigned membership in the wrong class is quite small for certain choices of KINIT. In addition, the correctly classified samples were given

relatively higher membership in their known class than in other classes.

As a final comparison, consider the results of the Bayes classifier for the TWOCLASS data. Running a ten percent jacknife procedure(Taking ten percent of the samples as test data and the remaining as training data, classifying these and then repeating the procedure until all samples have been used as test samples.) and assuming equal apriori probabilities for both classes, the Bayes classifier misclassified twenty of the samples. Clearly, dependent on the value chosen for K, the fuzzy nearest neighbor classifier can perform as well as a Bayes classifier.

4.4.2 Nearest Neighbor Computational Requirements

The computational requirements of the crisp and fuzzy classifiers are now considered. The number of multiplications and additions required to classify a sample are considered. The parameters which influence the number of multiplications and additions required are as follows.

C ≡ number of classes

K ≡ number of neighbors used to classify

N ≡ number of labelled samples used

n ≡ number of features per sample vector

	Fuzzy	Crisp
Multiplications	nN+C(2K+1)	nN
Additions	2nN+K+2CK	2nN+2K+CK+C-1

Using the parameters for the IRIS data set and setting K=3, the following particular example is given.

C = 3

N = 149

n = 4

	Fuzzy	Crisp
Multiplications	617	596
Additions	1213	1209

As this example illustrates, there is little difference in the computational requirements of the crisp and fuzzy K-nearest neighbor algorithms.

4.4.3 Results of Nearest Prototype Classifiers

The 1-nearest prototype classifier in both the crisp and fuzzy versions are the quickest and simplest of the classifiers considered. The reason is as follows. In both versions of the 1-nearest prototype algorithm, an unknown sample is compared to one prototype per class as opposed to the K-nearest neighbor algorithms wherein an entire set of labelled samples representing each class must be compared before the "K" nearest are obtained. The results reported in Table 5 show that the fuzzy nearest prototype classifier and the crisp nearest prototype classifier produced equivalent results. But, by looking at the memberships of the misclassified samples in terms of the number with membership greater than 0.7 in the wrong class, given in Table 6, it is clear that these memberships do provide a useful measure of level of confidence of classification. Further, the number of correctly classified samples with

memberships in the range between 0.5 and 0.7 is small compared to the number of correctly classified samples. This means that most of the correctly classified samples have membership in the correct class greater than 0.7. Thus, we can be assured based on the memberships assigned that the samples are correctly classified.

4.4.4 Nearest Prototype Computational Requirements

The computational requirements of the two classifiers are examined below. The number of multiplications and additions required for classification of a sample is given in terms of the parameters defined below.

C ≡ number of classes

n ≡ number of features per vector

	Fuzzy	Crisp
Multiplications	C(2+n)	Cn
Additions	C(2n+1)	2Cn

As with the previous comparisons, a particular example is given using the IRIS data.

C = 3

n = 4

	Fuzzy	Crisp
Multiplications	18	12
Additions	27	24

As with the nearest neighbor classifiers, there is little difference in the computational requirements of the crisp and fuzzy 1-nearest prototype classifiers.

4.5 Conclusions

The fuzzy K-means algorithm considered is a viable alternative for use in clustering problems. While

considerable research concerning this algorithm has already been conducted, the role of the weighting factor has not been investigated sufficiently. The results reported above indicate that the effect of using values higher than two for the weighting factor deserves further investigation.

The fuzzy K-nearest neighbor and fuzzy 1-nearest prototype algorithms developed and investigated in this report show useful results. In particular, concerning the fuzzy K-nearest neighbor algorithm with fuzzy k-nearest neighbor labelled sample membership assignments, the membership assignments produced for classified samples tend to possess desirable qualities. That is, an incorrectly classified sample will not have a membership in any class close to one while a correctly classified sample does possess a membership in the correct class close to one. The fuzzy 1-nearest prototype classifier, while not producing error rates as low as the fuzzy nearest neighbor classifier, also seems to produce membership assignments which are desirable.

Clearly, the results reported herein indicate that the fuzzy pattern recognition algorithms considered in this research are useful and should be further investigated.

Table 1

CLUSTERING RESULTS - IRIS data

Four Features

Fuzzy K-means

	1	2	3	
1	50	0	0	100%
2	0	47	3	94%
3	0	13	37	74%

Overall Correct rate 89.3%

Crisp K-means

	1	2	3	
1	50	0	0	100%
2	0	48	2	96%
3	0	14	36	72%

Overall correct rate 89.3%

Features Three and Four

Fuzzy K-means

	1	2	3	
1	50	0	0	100%
2	0	49	1	98%
3	0	7	43	86%

Overall correct rate 94.7%

Crisp K-means

	1	2	3	
1	50	0	0	100%
2	0	48	2	96%
3	0	7	43	86%

Overall correct rate 94.0%

Table 1a

CLUSTERING RESULTS - TWOCLASS data

Four Features

Fuzzy K-means				Crisp K-means			
	1	2			1	2	
1	114	7	94.2%	1	114	7	94.2%
2	15	106	87.6%	2	15	106	87.6%
Overall correct rate 90.9%				Overall correct rate 90.9%			

Features Three and Four

Fuzzy K-means				Crisp K-means			
	1	2			1	2	
1	114	7	94.2%	1	114	7	94.2%
2	15	106	87.6%	2	15	106	87.6%
Overall correct rate 90.9%				Overall correct rate 90.9%			

Table 1b

CLUSTERING RESULTS - LANDSAT data(four features)

Fuzzy K-means

	1	2	3	4	
1	11151	2951	2728	193	65.5%
2	3178	5054	723	11	56.4%
3	404	804	3429	14	73.3%
4	157	43	413	762	55.4%

Overall correct rate 63.7%

Crisp K-means

	1	2	3	4	
1	14267	1463	315	978	83.3%
2	5904	1575	1309	178	17.5%
3	2370	440	1481	363	31.8%
4	1254	30	38	53	0.3%

Overall correct rate 54.3%

SEARCH

	1	2	3	4	
1	9143	1165	230	3	87%
2	2088	3503	59	0	62%
3	380	1158	1409	0	48%
4	38	155	267	368	44%

Overall correct rate 72%

Table 2

Result of Varying the Weighting Factor(m)

	Vectors misclassified		Number of vector's membership in range (in class of maximum membership)							
			> 0.6		> 0.7		> 0.8		> 0.9	
m	T	I	T	I	T	I	T	I	T	I
1.4	22	17	18	15	13	14	8	12	4	7
1.5	22	17	17	15	11	12	5	9	2	5
1.6	22	17	17	14	9	12	4	6	1	1
1.7	22	17	16	14	7	10	2	5	1	0
1.8	22	17	16	13	5	8	2	2	0	0
1.9	22	16	14	13	4	5	1	0	0	0
2.0	22	16	10	8	4	5	1	0	0	0
2.1	22	16	9	8	2	2	1	0	0	0
2.2	22	16	9	8	2	0	0	0	0	0
2.3	22	16	8	6	2	0	0	0	0	0
2.4	21	15	7	6	2	0	0	0	0	0
2.5	21	15	7	4	1	0	0	0	0	0
2.6	21	15	6	3	1	0	0	0	0	0
2.7	21	15	4	1	1	0	0	0	0	0
2.8	21	15	4	0	1	0	0	0	0	0
2.9	20	15	4	0	0	0	0	0	0	0
3.0	20	15	4	0	0	0	0	0	0	0

Abbreviations: T-TWOCLASS data I-IRIS data

Table 2a

Result of Varying the Weighting Factor(m)

	Vectors misclassified		Number of vector's membership in range (in class of maximum membership)							
			> 0.6		> 0.7		> 0.8		> 0.9	
m	T	I	T	I	T	I	T	I	T	I
3.1	20	15	3	0	0	0	0	0	0	0
4.0	20	14	1	0	0	0	0	0	0	0
6.0	19	13	0	0	0	0	0	0	0	0
7.0	19	13	0	0	0	0	0	0	0	0
8.0	19	13	0	0	0	0	0	0	0	0
10.0	19	12	0	0	0	0	0	0	0	0
20.0	19	11	0	0	0	0	0	0	0	0
30.0	19	11	0	0	0	0	0	0	0	0

Abbreviations: T-TWOCLASS data I-IRIS data

Table 3

Results of K-nearest Neighbor Classifiers

Number of Misclassified vectors

	Crisp			Fuzzy-(1)			Fuzzy-(2)		Fuzzy-(3)		
K	I	T	I'	I	T	I'	T	I'	I	T	I'
1	6	26	6	6	26	6	26	6	6	26	6
2	7	26	7	6	26	6	21	6	6	21	6
3	6	21	6	6	22	6	21	7	5	19	6
4	5	20	5	6	19	6	20	7	5	20	5
5	5	20	5	5	21	5	20	7	4	19	4
6	6	19	6	5	18	5	20	6	4	20	4
7	5	19	5	5	21	5	18	6	4	19	4
8	7	21	7	6	18	6	20	6	4	20	4
9	6	21	6	4	21	4	18	5	4	18	4

Notation and Abbreviations: K-number of neighbors used
I-IRIS data(four features)
T-TWOCLASS data(four features)
I'-IRIS23 data(four features)
(1)-crisp initialization
(2)-exponential initialization
(3)-fuzzy 3-nearest neighbor initialization

Table 4

Results of Fuzzy K-nearest neighbor classifier,

with fuzzy KINIT-nearest neighbor initialization

	KINIT									
	1		3		5		7		9	
K	I	T	I	T	I	T	I	T	I	T
1	6-3	26-15	6-4	26-17	6-4	26-18	6-5	26-18	6-5	26-18
2	6-4	23-17	6-4	21-13	6-4	23-14	6-4	22-13	6-4	22-11
3	5-4	20-12	5-4	19-12	5-4	21-12	5-5	21-10	6-5	23-10
4	5-4	17-12	5-4	20-11	5-4	19-10	5-4	19-10	5-4	19-9
5	4-4	16-11	4-4	19-11	5-4	19-10	5-3	20-11	5-3	19-10
6	4-4	20-10	4-4	20-11	4-4	20-11	4-3	21-9	4-3	20-8
7	4-3	17-9	4-4	19-10	4-3	20-9	4-3	20-8	4-3	20-8
8	4-3	17-9	4-3	20-9	4-2	20-9	4-2	20-8	4-2	20-8
9	4-3	18-8	4-3	18-8	4-2	21-8	4-2	21-9	4-2	21-8

Abbreviations: I - IRIS data(four features)

T - TWOCLASS data(four features)

Note 1: Columns give results for the values of KINIT(the K used to initialize the labelled samples memberships) shown. Rows give results for values of K in the K-nearest neighbor algorithm

Note 2: Table entries are interpreted as: X-Y indicates X misclassified vectors with Y of the X given membership in the wrong class greater than 0.7..

Table 5

Results of the 1-Nearest Prototype Classifier

IRIS data

Four Features

	Crisp			Fuzzy			
	1	2	3		1	2	3
1	50	0	0	1	50	0	0
2	0	45	5	2	0	45	5
3	0	7	43	3	0	7	43

Features Three and Four

	Crisp			Fuzzy			
	1	2	3		1	2	3
1	50	0	0	1	50	0	0
2	0	48	2	2	0	48	2
3	0	4	46	3	0	4	46

TWOCLASS data

Four Features

	Crisp			Fuzzy			
	1	2			1	2	
1	113	8		1	113	8	
2	12	109		2	12	109	

Features Three and Four

	Crisp			Fuzzy			
	1	2			1	2	
1	113	8		1	113	8	
2	12	109		2	12	109	

Table 6

Fuzzy Classifier Membership Assignments

	IRIS data		TWOCLASS data	
	A	B	A	B
Misclassified samples with membership assigned > 0.7	1	1	3	3
Samples with membership assigned > 0.5 and < 0.7	15	15	36	36

Abbreviations: A - Four feature used

B - Features three and four used

Note: The first row in the table above gives the number of misclassified vectors in the indicated range and the second row gives the number of all classified samples in the given range. The intent is to illustrate that very few samples are misclassified with high membership, while very few correctly classified samples are given membership in their class in the "fuzzy" region between 0.5 and 0.7.

REFERENCES

1. Tou, Julius T., and Gonzalez, Rafael C., Pattern Recognition Principles, Reading, Massachussetts, Addison-Wesley, 1981

2. Duda, Richard O., and Hart, Peter E., Pattern Classification and Scene Analysis, New York, New York, Wiley and Sons, 1973

3. Gonzalez, Rafael C., and Thomason, Michael G., Syntactic Pattern Recognition: An Introduction, Reading, Massachussetts, Addison-Wesley, 1978

4. Zadeh, Lofti A., "Fuzzy Sets", Information and Control, Vol. 8, pp. 338-353, 1965

5. Bezdek, James C., Pattern Recognition with Fuzzy Objective Function Algorithms, New York, New York, Plenum Press, 1981.

6. Kaufmann, A., Introduction to the Theory of Fuzzy Subsets, Vol. I, New York, New York, Academic Press, 1975.

7. Kandel, Abraham, Fuzzy Techniques in Pattern Recognition, New York, New York, Wiley and Sons, 1982.

8. Dunn, J. C., "A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact Well-Separated Clusters", Journal of Cybernetics, Vol 3, pp 32-57.

9. Bezdek, James C., "A Physical Interpretation of Fuzzy ISODATA," IEEE Transactions on Systems, Man, and Cybernetics, SMC-6, pp. 387-389.

10. Dunn, J. C., "Some Recent Investigations of a New Fuzzy Partitioning Algorithm and its Application to Pattern Classification Problems", Journal of Cybernetics, Vol. 4, pp.1-15, 1974.

11. Anderson, E., "The Irises of the Gaspe Peninsula", Bulletin American Iris Society, Vol 59, pp. 2-5, 1935.

12. Honboonherm, Saowaluck, "Contextual Classification of Remotely Sensed Data", Phd Dissertation, University of Missouri, Columbia, Missouri, 157 pp., 1983.

13. Hunt, Doug J., "Incorporating Fuzzy Membership Functions into the Perceptron Pattern Recognition Algorithm', Technical Report, University of Missouri, Columbia, Missouri, 29 pp. 1984.

APPENDIX A

PROGRAM LISTINGS

This appendix includes the documented source code which was implemented for the research.

```
$OPTIMIZE
C   A DRIVER ROUTINE WHICH ALLOWS SELECTION OF ONE OF THREE
C   DATA SETS, AND THEN SELECTION OF ONE OF SEVEN PATTERN
C   RECOGNITION ALGORITHMS TO BE RAN ON THE DATA SET
C   CHOSEN. AFTER THE CHOSEN ALGORITHM HAS COMPLETED
C   THE USER IS PROMPTED FOR CHOICES OF INTERPRETIVE
C   ALGORITHMS TO RUN USING RESULTS PRODUCED BY
C   THE PATTERN RECOGNITION ALGORITHM CHOSEN.
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: JUNE 84
C
C   FILENAME: MGFUZZY.FTN
C
C   CALLING SEQUENCE: RUN MGFUZZY
C
C   DATA FILES AVAILABLE:
C
C    IRIS.DAT - 150 SAMPLES WITH FOUR FEATURES PER SAMPLE,
C               50 SAMPLES PER CLASS
C
C    TWOCLASS.DAT - 242 SAMPLES WITH FOUR FEATURES PER
C                   SAMPLE, 141 SAMPLES PER CLASS
C
C    IRIS23.DAT - 100 SAMPLES WITH FOUR FEATURES PER SAMPLE,
C                 50 SAMPLES PER CLASS( A SUBSET OF IRIS.DAT)
C
C   PATTERN RECOGNITON SUBROUTINES AVAILABLE ( BY TYPE ):
C
C       DESCRIPTION                              FILENAME
C_____________________________________________________________
C
C    1 - CRISP K-MEANS                          CRSKMEAN.FTN
C    2 - FUZZY K-MEANS                          FUZKMEAN.FTN
C    3 - FUZZY K-NEAREST NEIGHBOR               FUZNEARN.FTN
C    4 - FUZZY 1-NEAREST PROTOTYPE              FUZPROTO.FTN
C    5 - FUZZY K-NEAREST NEIGHBOR;              FUZNEARN.FTN
C         FUZZY INITIALIZATION
C        A - VIA "FZIFY" ;                      MGFZIFY.FTN
C             DEVELOPED BY D. HUNT
C        B - VIA "FZFYNN";                      MGFZFYNN.FTN
C             A NEAREST NEIGHBOR TECHNIQUE
C    6 - CRISP K-NEAREST NEIGHBOR               CRSIPNN.FTN
C    7 - CRISP 1-NEAREST NEIGHBOR               CRISPNP.FTN
C
C     NOTE: CLASSIFIER ALGORITHMS (3 THROUGH 7) USE
C           THE "LEAVE ONE OUT" METHOD TO PRODUCE A
C           CLASSIFICATION.  THE METHOD IS SIMPLY TO
C           LEAVE THE CURRENT SAMPLE BEING CLASSIFIED
C           OUT OF THE LABELLED SET USED TO
C           DETERMINE CLASSIFICATION.
C
C
C   INTERPRETATIVE SUBROUTINES AVAILABLE:
C
C         DESCRIPTION                              FILENAME
C_____________________________________________________________
C
C    1 - COMPUTE HARD PARTITION USING           MGCMTRIX.FTN
C        MAXIMUM CLASS MEMBERSHIP,
C        RESULT IS A CONFUSION MATRIX
C    2 - OUTPUT THE MEMBERSHIP FUNCTIONS,       MGMEMBPR.FTN
C        A - FOR ALL SAMPLES OF DATA SET
C        B - FOR ONLY THE MISCLASSIFIED
C            SAMPLES OF DATA SET
C    3 - COMPUTE LEVEL SETS USING ALPHA         MGCUTSET.FTN
C        AND BETA AS UPPER AND LOWER
C        CUTOFFS, RESPECTIVELY. BASICALLY,
C        A LEVEL SET IS DEFINED TO INCLUDE
C        THOSE SAMPLES WHICH HAVE MEMBERSHIP
C        IN THE DESIRED RANGE, EITHER GREATER
C        THAN ALPHA, LESS THAN BEAT, OR IN-
C        BETWEEN ALPHA AND BETA
```

```
C    4 - OUTPUT A 2-D PLOT OF RESULTS,          MGDOPLOT.FTN
C        A - RESULTANT CLASSIFICATION OF
C            EACH SAMPLE BY CLASS
C        B - MEMBERSHIP FUNCTIONS ASSIGNED
C            IN ONE CLASS
C        C - SAMPLES WHICH ARE A LEVEL SET
C            FOR A GIVEN CLASS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C    LOGICAL UNITS USED:
C
C          11 - TO ACCESS A DATA FILE, OPEN ONLY WHEN
C               ACCESSING THE DATA SET AND THEN NO
C               LONGER USED UNTIL ANOTHER DATA FILE
C               IS SELECTED
C
C          5 - USED TO READ AND WRITE TO CONSOLE,
C              ALWAYS ASSIGNED
C
C          6 - USED TO WRITE TO HARD COPY PRINTER,
C              ALWAYS ASSIGNED, THOUGH ONLY USED
C              WHEN USER SELECTS THE PRINTER AS
C              THE OUTPUT DEVICE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   FINAL NOTE: USER INTERACTION IS REQUIRED AT
C               VARIOUS INTERVALS. AFTER A PARTICULAR
C               PATTERN RECOGNITION ALGORITHM IS CHOSEN
C               NO FURTHER INTERACTION IS REQUIRED OTHER
C               THAN TO CHOOSE THE VALUE FOR "K" IF A
C               NEAREST NEIGHBOR ALGORITHM, OR CHOOSE THE
C               TYPE OF "FUZZY" INITIALIZATION IF SO CHOSEN
C               UNTIL THE ALGORITHM HAS COMPLETED AND
C               INTERPRETIVE OPTIONS BECOME AVAILABLE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      LOGICAL DONE
      CHARACTER*2 REPLY
      CHARACTER*15 VFORMT,DFILE
      REAL MFUNCT(3,242),PROTO(4,3),X(4,242)
      REAL NMFUNC(3,242),LOW
      INTEGER VECSIZ,VCOUNT,FFEAT,CLASS,VECTOR,CHOICE
      INTEGER FETUR,CHSIZE,VCLASS(3),TEST1,START(3)
      INTEGER ACOUNT(3),BCOUNT(3),ICOUNT(3),OPTION,POINTR
      INTEGER WRNCNT,KFINIT,FUZCH,VECREC,COUNT,END(3)
      INTEGER*2 KALPHA(3,242),KBETA(3,242),BETWEN(3,242)
      INTEGER*2 WRONG(242)
      COMMON /AREA1/X /AREA2/PROTO /AREA3/MFUNCT
      COMMON /AREA4/WRNCNT,WRONG
      COMMON /AREA5/NMFUNC /AREA6/START,END
      COMMON /AREA8/ACOUNT,BCOUNT,ICOUNT,KALPHA,
     1         KBETA,BETWEN
C
C  *** EXECUTION BEGINS ***
C
C     LOOP(Until user is finished)
1     CONTINUE
C
C    FIND OUT WHICH DATA SET TO USE
C
      WRITE(5,2)
2     FORMAT(//,' Enter code for data set to use:',
     1       //,5X,'1-IRIS data-150 vectors: 3 classes-'
     2       '4 features',//,5X,'2-TWOCLASS data-242'
     3       ' vectors: 2 classes-4 features',//5X,
     4       '3-IRIS23 data-100 vectors: 2 classes-'
     5       '4 features',/,7X,'(This file contains'
     6       ' classes two and three of IRIS)')
      READ(5,6) CHOICE
6     FORMAT(I1)
```

```
C
C
C       DO CASE(CHOICE)
        GO TO (10,15,20) CHOICE
C
C        CASE #1
10       CONTINUE
         DFILE='IRIS.DAT'
         VFORMT='(20F3.1)'
         GO TO 30
C        END CASE #1
C
C        CASE #2
15       CONTINUE
         DFILE='TWOCLASS.DAT'
         VFORMT='(4F10.6)'
         GO TO 30
C        END CASE #2
C
C        CASE #3
20       CONTINUE
         DFILE='IRIS23.DAT'
         VFORMT='(20F3.1)'
         GO TO 30
C        END CASE #3
C
30      CONTINUE
C       END DO CASE
C
C   OPEN DATA FILE AND READ IN NUMBER OF CLASSES,
C   SIZE OF DATA VECTORS, NUMBER OF VECTORS PER
C   RECORD, AND NUMBER OF VECTORS PER CLASS.
C
        OPEN(11,FILE=DFILE)
C
        READ(11,31) KLASES,VECSIZ,VECREC
31      FORMAT(3I3)
        READ(11,31) (VCLASS(I),I=1,KLASES)

        VCOUNT=0
C
C       DO UNTIL(REMAINING DATA SET DEPENDENT
C                VARIABLES INITIALIZED)
        DO 32 CLASS=1,KLASES
C
          VCOUNT=VCOUNT+VCLASS(CLASS)
          END(CLASS)=VCOUNT
          START(CLASS)=END(CLASS)-VCLASS(CLASS)+1
C
32     CONTINUE
C      END DO UNTIL
C
C    READ DATA VECTORS FROM DISK FILE,
C     THEN CLOSE THE DATA FILE
C
C      DO UNTIL(DATA VECTORS READ)
       DO 33 I=1,VCOUNT,VECREC
         READ(11,VFORMT) ((X(J,L),J=1,VECSIZ),
     1 L=I,I+VECREC-1)
33     CONTINUE
C      END DO UNTIL
C
       CLOSE(11)
C
C      LOOP(UNTIL ANOTHER DATA SET IS DESIRED)
34     CONTINUE
C
C
C    LET USER KNOW HOW MANY FEATURES ARE AVAILABLE
C
         WRITE(5,35) VECSIZ
35       FORMAT(/,4X,'There are',I2,' features in each '
     1 'vector.')
C
```

```
C    SET FIRST FEATURE AND LAST FEATURE TO BE CONSIDERED
C    WHEN COMPUTING MEMBERSHIP FUNCTION ARRAY.
        WRITE(5,40)
40      FORMAT(//,' Input first feature number in vectors'
     1 'to use(I1).')
        READ(5,*) FFEAT
        WRITE(5,45)
45      FORMAT(//,' Input last feature number in vectors'
     1 'to use(I1).')
        READ(5,*) LFEAT
C
C    INPUT VALUE OF "FUZZIFIER" TO BE USED IN ALGORITHM
C
        WRITE(5,50)
50      FORMAT('1',' Input value of weighting factor'
     1 '("FUZZIFIER"):')
        READ(5,55) FZFIER
55      FORMAT(F3.1)
C
C    TEST IF VALUE OF "FUZZIFIER" IN PROPER RANGE
C     (ONLY ALLOW > 1.3)
C
        IF(.NOT.(FZFIER.LT.1.3))GO TO 60
         FZFIER=1.3
C       END IF
60      CONTINUE
C
C    SET MAXIMUM MEMBERSHIP UPDATE ERROR ALLOWED
C    IN FUZZY K-MEANS ALGORITHM
C
        EPSLON=0.001
C
C     FIND OUT WHAT TO DO
C
        WRITE(5,65)
65      FORMAT(//,' Enter code for your choice:',
     1        //,5X,'1 - crisp K-means',
     2        //,5X,'2 - fuzzy K-means',
     3        //,5X,'3 - fuzzy K-nearest neighbor',
     4        //,5X,'4 - fuzzy 1-nearest prototype',
     5        //,5X,'5 - fuzzy K-nearest neighbor,'
     6' fuzzy initialization',
     7        //,5X,'6 - crisp K-nearest neighbor'
     8        //,5X,'7 - crisp 1-nearest prototype')
        READ(5,6) CHOICE
C
        GO TO (90,95,100,100,100,100,100) CHOICE
C
C        CASE #1
90       CONTINUE
C
C   INITIALIZE MEMBERSHIP FUNCTIONS,
c    THEN RUN CRISP K-MEANS ALGORITHM
C
         HIGH=1.0
         LOW=0.0
         CALL INITMF(VCOUNT,FFEAT,LFEAT,KLASES,HIGH,LOW)
         CALL CRMEAN(KLASES,VCOUNT,FFEAT,LFEAT,EPSLON,
     1                ITERAT)
         GO TO 400
C        END CASE #1
C
C        CASE #2
95       CONTINUE
C
           HIGH=0.98
           LOW=0.02
           CALL INITMF(VCOUNT,FFEAT,LFEAT,KLASES,HIGH,LOW)
           CALL FKMEAN(FZFIER,KLASES,FFEAT,LFEAT,VCOUNT,
     1                 EPSLON,ITERAT)
         GO TO 400
C        END CASE #2
C
```

```
C       CASES #3,4,5&6
100     CONTINUE
        IF((CHOICE.EQ.3).OR.(CHOICE.EQ.5).OR.
     1                                (CHOICE.EQ.6)) THEN
          WRITE(5,105)
105       FORMAT(/,' Input number of neighbors used to '
     1    'assign membership function values(<10):')
          READ(5,6) K
          IF(CHOICE.EQ.5) THEN
            WRITE(5,106)
106         FORMAT(/,' Input number of neighbors used to'
     1      ' fuzzify memberships of labelled set(<10):')
            READ(5,6) KFINIT
            WRITE(5,107)
107         FORMAT(/,' Enter choice of fuzzifying:',//,
     1         ' 1 - Fuzzifying per nearest neighbor(s)'
     2         'technique',//,' 2 - Fuzzifying per D.'
     3         'HUNT technique(two class sets only)')
            READ(5,6) FUZCH
          END IF
        END IF

        IF((CHOICE.NE.4).AND.(CHOICE.NE.7)) THEN
C
C  INITIALIZE THE MEMBERSHIP FUNCTIONS
C
C         DO UNTIL(ALL MEMBERSHIP FUNCTIONS SET TO ZERO)
          DO 115 CLASS=1,KLASES
            DO 110 VECTOR=1,VCOUNT
              MFUNCT(CLASS,VECTOR)=0.0
110         CONTINUE
115       CONTINUE
C         END DO UNTIL
C
C         DO UNTIL(ALL DATA ASSIGNED COMPLETE
c                 MEMBERSHIP IN THEIR CLASS)
          DO 125 CLASS=1,KLASES
            DO 120 VECTOR=START(CLASS),END(CLASS)
              MFUNCT(CLASS,VECTOR)=1.0
120         CONTINUE
125       CONTINUE
C         END DO UNTIL
C
        END IF
C
C  SET FIRST VECTOR IN DATA SET AS THE
C   FIRST TEST VECTOR
C
C       DO UNTIL(TRAINING PROCEDURE COMPLETED)
        DO 300 TEST1=1,VCOUNT
C
          IF((CHOICE.EQ.3).OR.(CHOICE.EQ.5)) THEN
            IF(CHOICE.EQ.5) THEN
              IF(FUZCH.EQ.2) THEN
                CALL FUZFY(FZFIER,FFEAT,LFEAT,VCOUNT)
              ELSE
                CALL FZFYNN(FZFIER,KLASES,VCOUNT,FFEAT,
     1                      LFEAT,KFINIT,TEST1)
              END IF
            END IF
            CALL FUZNN(FZFIER,KLASES,FFEAT,LFEAT,VCOUNT,
     1                                     K,TEST1)
          ELSE IF((CHOICE.EQ.4).OR.(CHOICE.EQ.7)) THEN
C
C           DO UNTIL(PROTOTYPES FOUND FOR ALL CLASSES)
            DO 230 CLASS=1,KLASES
C
C             DO UNTIL(PROTOTYPE VECTOR FOR CURRENT
C                     CLASS ZEROED)
              DO 205 FETUR=FFEAT,LFEAT
                PROTO(FETUR,CLASS)=0.0
205           CONTINUE
C             END DO UNTIL
```

```
C
C               DO UNTIL(PROTOTYPE FOR CURRENT CLASS SUMMED)
                DO 220 VECTOR=START(CLASS),END(CLASS)
                  IF(VECTOR.NE.TEST1) THEN
                    DO 210 FETUR=FFEAT,LFEAT
                      PROTO(FETUR,CLASS)=PROTO(FETUR,CLASS)
     1                                   +X(FETUR,VECTOR)
210                 CONTINUE
                  END IF
220               CONTINUE
C               END DO UNTIL
C
                IF((TEST1.GE.START(CLASS)).AND.
     1             (TEST1.LE.END(CLASS))) THEN
                  COUNT=VCLASS(CLASS)-1
                ELSE
                  COUNT=VCLASS(CLASS)
                END IF
C
C               DO UNTIL(PROTOTYPE DIVIDED BY COUNT
C                     OF TRAINING SET)
                DO 225 FETUR=FFEAT,LFEAT
                  PROTO(FETUR,CLASS)=PROTO(FETUR,CLASS)/
     1                                       COUNT
225               CONTINUE
C               END DO UNTIL
C
230           CONTINUE
C             END DO UNTIL
C
              IF(CHOICE.EQ.4) THEN
                CALL FPROTO(FZFIER,KLASES,FFEAT,LFEAT,TEST1)
              ELSE
                CALL CRSPNP(KLASES,FFEAT,LFEAT,TEST1)
              END IF
            ELSE IF(CHOICE.EQ.6) THEN
C
              CALL CRSPNN(KLASES,FFEAT,LFEAT,K,VCOUNT,TEST1)
C
            END IF
C
300       CONTINUE
C         END DO UNTIL
          GO TO 400
C         END CASE # 3, 4, 5, 6 & 7
C
400     CONTINUE
C       END DO CASE
C
C   *** OUTPUT AND INTERPRET RESULTS ***
C
        WRITE(5,500)
500     FORMAT(//,' Where would you like results sent?'
       1' Enter choice:',
       2      //,5X,'5 - CONSOLE',
       3      //,5X,'6 - PRINTER')
        READ(5,506) LU
506     FORMAT(I1)
C
        WRITE(LU,507) DFILE,K,KFINIT
507     FORMAT(' ',' Data set: ',A20,' K=',I3,'KFINIT=',I3)
C
        IF((CHOICE.EQ.3).OR.(CHOICE.EQ.5).OR.
     1                      (CHOICE.EQ.6)) THEN
          WRITE(LU,508) K
508       FORMAT(' ',' Number of neighbors used = ',I3)
          IF((CHOICE.EQ.5).AND.(FUZCH.EQ.1)) THEN
            WRITE(LU,509) KFINIT
509         FORMAT(' ','Number of neighbors used
       1'for initialization = ',I3)
          END IF
        END IF
C
```

```
      IF((CHOICE.EQ.1).OR.(CHOICE.EQ.2)) THEN
C
C  OUTPUT "EPSLON", AND NUMBER OF ITERATIONS REQUIRED
C
         WRITE(LU,602) EPSLON,ITERAT
602      FORMAT(' ','EPSLON (maximum update error allowed'
     1 ) = ',F7.5,' .',I3,' iterations required.')
C
      END IF
C
      IF((CHOICE.GE.2).AND.(CHOICE.LE.5)) THEN
C
C  OUTPUT FUZZIFIER
C
         WRITE(LU,603) FZFIER
603      FORMAT(' ',' The weighting factor("FUZZIFIER") is:'
     1          ,F6.3,' .')
C
      END IF
C
      IF((CHOICE.NE.3).AND.(CHOICE.NE.5).AND.
     1                      (CHOICE.NE.6)) THEN
C
C  OUTPUT THE FINAL CLUSTER CENTERS
C
         DO 615 INDEX=1,KLASES
           WRITE(LU,612) INDEX,(PROTO(I,INDEX),
     1     I=FFEAT,LFEAT)
612        FORMAT(/,1X,'Weighted mean for class #',
     1     I2,' :',12F6.3)
615      CONTINUE
C
      END IF
C
C  OUTPUT A CONFUSION MATRIX AND FIND THE INDICES
C  OF MISCLASSIFIED VECTORS
C
      CALL CMTRIX(KLASES,VCOUNT,LU,CHOICE)
C
      IF((CHOICE.GE.2).AND.(CHOICE.LE.5)) THEN
C
C  OUTPUT THE MEMBERSHIP FUNCTION ARRAY COMPUTED
C
         WRITE(5,669)
669      FORMAT(/,' Enter option:',//,
     1     5X,'1 - Output entire membership function '
     2 'array',//,5X,'2 - Output only the misclassified'
     3     'vector"s membership functions')
         READ(5,506) OPTION
C
         CALL MEMBPR(KLASES,VCOUNT,LU,CHOICE,OPTION)
C
      END IF
C
C      LOOP(UNTIL USER FINISHED)
670    CONTINUE
C
         WRITE(5,969)
969      FORMAT(/,' Do you want to find the ALPHA and'
     1'BETA cutsets?(Y/N)')
         READ(5,996) REPLY
         IF(REPLY.EQ.'Y') THEN
C
           WRITE(5,971)
971        FORMAT(' ','Input "ALPHA", upper membership'
     1     ' cut-off:')
           READ(5,972) ALPHA
972        FORMAT(F4.3)
           WRITE(5,973)
973        FORMAT(' ','Input "BETA", lower non-membership'
     1     ' cut-off:')
           READ(5,972) BETA
C
```

```
          CALL CUTSET(ALPHA,BETA,KLASES,VCOUNT,CHOICE,LU)
C
        END IF
C
        WRITE(5,995)
995     FORMAT(/,' Do you want a plot of results?(Y/N)')
        READ(5,996) REPLY
996     FORMAT(A1)
C
        IF(REPLY.EQ.'Y') THEN
          CALL DOPLOT(KLASES,VCOUNT,START,END,LFEAT,CHOICE)
        END IF
C
        WRITE(5,997)
997     FORMAT(/,' Do you want another plot or cutset'
     1      '?(Y/N)')
        READ(5,996) REPLY
       IF(REPLY.EQ.'Y')GO TO 670
C      END LOOP
C
      WRITE(5,998)
998   FORMAT(//,' Do you want to run another algorithm'
     1' using the same data set?(Y/N)')
      READ(5,996) REPLY
      IF(REPLY.EQ.'Y')GO TO 34
      WRITE(5,999)
999   FORMAT(//,' Do you want to get a different'
     1'data set?(Y/N)')
      READ(5,996) REPLY
      IF(REPLY.EQ.'Y')GO TO 1
C     END LOOP
C
        STOP
        END
```

```
$OPTIMIZE
C  A SUBROUTINE WHICH IMPLEMENTS THE FUZZY K-MEANS
C  ALGORITHM( ALSO REFERRED TO AS THE FUZZY
C  ISODATA ALGORITHM).
C
C  WRITTEN BY: MICHAEL R. GRAY
C
C  COMPLETED: MAR 1984
C
C  FILENAME: FUZKMEAN.FTN
C
C  CALLING SEQUENCE(FROM A FORTRAN ROUTINE): CALL
C  FKMEAN(FZFIER,KLASES,FFEAT,LFEAT,VCOUNT,EPSLON,ITERAT)
C
C  INPUT VARIABLES(NOT CHANGED):
C
C   FZFIER - REAL VALUE FOR THE WEIGHTING FACTOR(M)
C            USED BY FUZZY K-MEANS
C
C   KLASES - INTEGER COUNT OF NUMBER OF CLUSTER DESIRED
C
C   FFEAT - INTEGER WHICH SETS FIRST FEATURE IN
C            DATA SET TO CONSIDER
C
C   LFEAT - INTEGER WHICH SETS LAST FEATURE IN
C            DATA SET TO CONSIDER
C
C   VCOUNT - INTEGER COUNT OF SAMPLES IN DATA SET
C
C   EPSLON - REAL VALUE WHICH SETS THE MAXIMUM UPDATE
C            ERROR ALLOWED IN ANY MEMBERSHIP
C            ASSIGNMENT BEFORE COMPLETION
C
C   X - REAL ARRAY 4 BY 242 WHICH HOLDS THE DATA SAMPLES,
C       PASSES IN LABELLED FORTRAN COMMON "AREA1"
C
C  OUTPUT VARIABLES PRODUCED:
C
C   PROTO - REAL ARRAY 4 BY 3 WHICH HOLDS THE FUZZY
C           CLUSTER CENTERS PRODUCED, PASSED
C           IN LABELLED FORTRAN COMMOM "AREA2"
C
C   MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C            MEMBERSHIP FUNCTION ASSIGNMENTS PRODUCED,
C            PASSED IN LABELLED FORTRAN COMMON "AREA3"
C
C   ITERAT - INTEGER WHICH HOLDS THE NUMBER OF
C            ITERATIONS REQUIRED
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC             CCCCC
C
C                        PSEUDO - CODE SOLUTION
C
C  ENTER FUZKMEAN
C   SET MAXIMUM NUMBER OF ITERATIONS ALLOWED
C   INITIALIZE ITERATIONS TO 0
C   DO UNTIL(MEMBERSHIP FUNCTIONS ASSIGNED STABILIZE
C            (UPDATE ERROR OF ANY MEMBERSHIP
C            ASSIGNMENT < EPSILON OR MAXIMUM
C             NUMBER OF ITERATIONS COMPLETE)
C     COMPUTE FUZZY CLUSTER CENTERS BASED ON
C      MEMBERSHIP ARRAY INITIALIZATION
C     SET VECTOR INDEX=1
C     DO UNTIL(ALL DATA VECTORS ASSIGNED
C              MEMBERSHIP FUNCTIONS)
C       COMPUTE DISTANCES FROM EACH FUZZY CLUSTER
C       CENTER TO CURRENT DATA VECTOR
C       ASSIGN DATA VECTOR MEMBERSHIPS IN ALL CLASSES
C        AS A FUNCTION OF DISTANCE FROM FUZZY CLUSTER
C        CENTER OF THE CLASS, KEEPING TRACK OF MAXIMUM
C        UPDATE DIFFERENCE FOR ALL MEMBERSHIP ASSIGNMENTS
C       INCREMENT VECTOR INDEX
C     END DO UNTIL
C     INCREMENT ITERATION COUNT
```

```
C   END DO UNTIL
C  RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE FKMEAN(FZFIER,KLASES,FFEAT,LFEAT,
     1                  VCOUNT,EPSLON,ITERAT)
C
      LOGICAL MATCH
      REAL MFUNCT(3,242),X(4,242),NEWMF,PROTO(4,3),XDIST(3)
      INTEGER FFEAT,LFEAT,CLASS,VECTOR,FETUR,VCOUNT
      COMMON /AREA1/X /AREA2/PROTO /AREA3/MFUNCT
C
C  *** INITIALIZE ***
C
C  SET MAXIMUM ITERATIONS
C
      MAXITR=50
C
C  SET ITERATION TO ZERO INITIALLY
C
      ITERAT=0
C
C  COMPUTE POWER WHICH IS A FUNCTION OF THE "FUZZIFIER"
C
      POWER=1.0/(FZFIER-1.0)
C
C  ***  BEGIN ITERATIONS  ***
C
C      DO UNTIL(MAXIMUM DIFFERENCE IN MEMBERSHIP
C               FUNCTION UPDATES LESS THAN EPSLON
C               OR MAXIMUM ITERATIONS PERFORMED)
1       CONTINUE
C
C  COMPUTE WEIGHTED MEANS
C
C       DO UNTIL(WEIGHTED MEAN FOR EACH CLASS COMPUTED)
        DO 7 CLASS=1,KLASES
C
C         DO UNTIL(MEAN VECTOR # "CLASS" ZEROED)
          DO 2 FETUR=FFEAT,LFEAT
            PROTO(FETUR,CLASS)=0.0
2         CONTINUE
C         END DO UNTIL
C
          DENOM=0.0
C
C         DO UNTIL(ALL VECTORS OF CLASS "TCLASS" INCLUDED)
          DO 4 VECTOR=1,VCOUNT
C
            FUZZED=MFUNCT(CLASS,VECTOR)**FZFIER
            DENOM=DENOM+FUZZED
C
C           DO UNTIL(VECTOR NUMBER "VECTOR" INCLUDED)
            DO 3 FETUR=FFEAT,LFEAT
              PROTO(FETUR,CLASS)=PROTO(FETUR,CLASS)+
     1                           X(FETUR,VECTOR)*FUZZED
3           CONTINUE
C           END DO UNTIL
C
4         CONTINUE
C         END DO UNTIL
C
C         DO UNTIL(MEAN VECTOR DIVIDED BY "DENOM")
          DO 6 FETUR=FFEAT,LFEAT
            PROTO(FETUR,CLASS)=PROTO(FETUR,CLASS)/DENOM
6         CONTINUE
C
7       CONTINUE
C       END DO UNTIL
C
          DIFMAX=0.0
C
C         DO UNTIL(MEMBERSHIP FUNCTIONS UPDATED AND
```

```
C                  MAXIMUM OLD TO NEW MEMBERSHIP
C                  FUNCTION DIFFERENCE FOUND FOR
C                  ALL MEMBERSHIP FUNCTIONS)
         DO 14 VECTOR=1,VCOUNT
C
            MATCH=.FALSE.
            DSUM=0.0
            CLASS=1
C
C           DO UNTIL(VECTOR "VECTOR" COMPARED TO ALL
C                    MEANS OR MATCH FOUND)
8           CONTINUE
C
C  COMPUTE DISTANCE - VECTOR NUMBER "VECTOR"
C   TO MEAN NUMBER "CLASS"
C
              XDIST(CLASS)=0.0
C
C             DO UNTIL(DISTANCE SQUARED COMPUTED)
              DO 9 FETUR=FFEAT,LFEAT
                TEMP=X(FETUR,VECTOR)-PROTO(FETUR,CLASS)
                XDIST(CLASS)=XDIST(CLASS)+TEMP*TEMP
9             CONTINUE
C             END DO UNTIL
C
              IF(XDIST(CLASS).EQ.0.0) THEN
                MATCH=.TRUE.
                MCLASS=CLASS
              ELSE
                XDIST(CLASS)=1.0/XDIST(CLASS)**POWER
                DSUM=DSUM+XDIST(CLASS)
              END IF
C
              CLASS=CLASS+1
C
            IF((.NOT.MATCH).AND.(CLASS.LE.KLASES))GO TO 8
C           END DO UNTIL
C
            IF(.NOT.MATCH) THEN
C             DO UNTIL(NEW MEMBERSHIP FUNCTIONS ASSIGNED,
C                      AND MAXIMUM OLD TO NEW MEMBERSHIP
C                      FUNCTION DIFFERENCE FOUND FOR
C                      CURRENT VECTOR)
              DO 10 CLASS=1,KLASES
                NEWMF=XDIST(CLASS)/DSUM
                DIFF=ABS(NEWMF-MFUNCT(CLASS,VECTOR))
                MFUNCT(CLASS,VECTOR)=NEWMF
C
C  FIND MAXIMUM OF ALL DIFFERENCES IN OLD TO
C   TO NEW MEMBERSHIP FUNCTIONS
C
                IF(DIFF.GT.DIFMAX) THEN
                  DIFMAX=DIFF
                END IF
C
10            CONTINUE
C             END DO UNTIL
C
            ELSE
C
C             DO UNTIL(NEW MEMBERSHIP FUNCTIONS ASSIGNED,
C                      AND MAXIMUM OLD TO NEW MEMBERSHIP
C                      FUNCTION DIFFERENCE FOUND FOR
C                      CURRENT VECTOR)
              DO 11 CLASS=1,KLASES
C
                IF(CLASS.EQ.MCLASS) THEN
                  NEWMF=1.0
                ELSE
                  NEWMF=0.0
                END IF
                DIFF=ABS(NEWMF-MFUNCT(CLASS,VECTOR))
                MFUNCT(CLASS,VECTOR)=NEWMF
C
```

```
C  FIND MAXIMUM OF ALL DIFFERENCES IN OLD
C  TO NEW MEMBERSHIP FUNCTIONS
C
                IF(DIFF.GT.DIFMAX) THEN
                  DIFMAX=DIFF
                END IF
C
11             CONTINUE
C              END DO UNTIL
             END IF
C
14         CONTINUE
C          END DO UNTIL
C
C  INCREMENT ITERATION COUNT
C
           ITERAT=ITERAT+1
C
          IF((DIFMAX.GT.EPSLON).AND.
     1                 (ITERAT.LE.MAXITR))GO TO 1
C         END DO UNTIL
C
          RETURN
          END
```

```
$OPTIMIZE
C  A SUBROUTINE WHICH IMPLEMENTS THE HARD K-MEANS
C  ALGORITHM USING MEMBERSHIP FUNCTIONS( =0,1 )
C  INSTEAD OF CLUSTER ASSIGNMENT.
C
C  WRITTEN BY: MICHAEL R. GRAY
C
C  COMPLETED: MAY 1984
C
C  FILENAME: CRSKMEAN.FTN
C
C  CALLING SEQUENCE(FROM A FORTRAN ROUTINE): CALL
C   CKMEAN(KLASES,VCOUNT,FFEAT,LFEAT,EPSLON,ITERAT)
C
C  INPUT VARIABLES(NOT CHANGED):
C
C   KLASES - INTEGER COUNT OF CLASSES,
C            OR CLUSTERS TO PRODUCE
C
C   VCOUNT - INTEGER COUNT OF SAMPLE VECTOR IN DATA SET
C
C   FFEAT - INTEGER WHICH SETS FIRST FEATURE IN DATA
C           VECTORS TO CONSIDER
C
C   LFEAT - INTEGER WHICH SETS LAST FEATURE IN DATA
C           VECTORS TO CONSIDER
C
C   EPSLON - REAL VALUE WHICH SETS MAXIMUM ERROR
C            ALLOWED IN CLUSTER UPDATE BEFORE
C            STOPPING ITERATIONS
C
C   X - REAL ARRAY 4 BY 242 WHICH HOLDS THE DATA SAMPLES,
C       PASSES IN LABELLED FORTRAN COMMON BLOCK "AREA1"
C
C  OUTPUT VARIABLES PRODUCED:
C
C   PROTO - REAL ARRAY 4 BY 3 WHICH HOLDS THE CLUSTER
C           CENTERS PRODUCED, PASSED IN LABELLED
C           FORTRAN COMMON BLOCK "AREA2"
C
C   MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C            MEMBERSHIP FUNCTION ASSIGNMENTS(0 OR 1)
C            PASSED IN LABELLED FORTRAN
C            COMMON BLOCK "AREA3"
C
C   ITERAT - INTEGER WHICH HOLDS THE NUMBER
C            OF ITERATIONS REQUIRED
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                      PSEUDO - CODE  SOLUTION
C
C   ENTER CRMEAN
C     SET MAXIMUM NUMBER OF ITERATIONS ALLOWED
C     COMPUTE INITIAL CLUSTER MEANS
C     INITIALIZE ITERATIONS TO 0
C     DO UNTIL(CLUSTER MEANS STABILIZE(UPDATE ERROR
C              < EPSILON OR MAXIMUM NUMBER
C              OF ITERATIONS COMPLETE)
C       SET VECTOR INDEX=1
C       DO UNTIL(ALL VECTORS IN DATA SET ASSIGNED
C                TO CLUSTER OF CLOSEST MEAN)
C         SET CLUSTER-MEAN INDEX=1
C         DO UNTIL(CLOSEST CLUSTER MEAN TO
C                  CURRENT VECTOR FOUND)
C           COMPUTE DISTANCE - CURRENT VECTOR
C            TO CURRENT CLUSTER MEAN
C           IF (FIRST CLUSTER MEAN IN LIST) THEN
C             SET MINIMUM DISTANCE TO DISTANCE COMPUTED
C             SET CLOSEST INDEX TO 1
C           ELSE IF (DISTANCE LESS THAN PREVIOUS
C                                    MINIMUM) THEN
C             SET MINIMUM DISTANCE TO NEW MINIMUM
C             SET CLOSEST INDEX TO THAT OF NEW MINIMUM
```

```
C            END IF
C            INCREMENT CLUSTER-MEAN INDEX
C          END DO UNTIL
C          ASSIGN CURRENT VECTOR TO CLUSTER OF CLOSEST MEAN
C          INCREMENT VECTOR INDEX
C        END DO UNTIL
C        COMPUTE NEW CLUSTER MEANS BASED ON NEW
C         ASSIGNMENT AND FIND MAXIMUM UPDATE
C         ERROR FOR ALL CLUSTER MEANS
C        INCREMENT ITERATION COUNT
C      END DO UNTIL
C    RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       SUBROUTINE CRMEAN(KLASES,VCOUNT,FFEAT,LFEAT,
     1                   EPSLON,ITERAT)
C
       REAL MFUNCT(3,242),X(4,242),PROTO(4,3),UPDATE(4)
       INTEGER CLOSER,VCOUNT,FFEAT,LFEAT,CLASS,VECTOR
       INTEGER FETUR,CCOUNT
       COMMON /AREA1/X /AREA2/PROTO /AREA3/MFUNCT
C
C   *** INITIALIZE ***
C
       MAXITR=50
C
C   COMPUTE CLUSTER MEANS BASES ON INITIAL
C    MEMBERSHIP ASSIGNMENTS
C
C        DO UNTIL(CLUSTER MEAN FOR EACH CLASS COMPUTED)
         DO 5 CLASS=1,KLASES
C
C          DO UNTIL(MEAN VECTOR # "CLASS" ZEROED)
           DO 1 FETUR=FFEAT,LFEAT
             PROTO(FETUR,CLASS)=0.0
1          CONTINUE
C          END DO UNTIL
C
           CCOUNT=0
C
C          DO UNTIL(MEAN FOR CURRENT CLASS COMPUTED)
           DO 3 VECTOR=1,VCOUNT
C
             IF(MFUNCT(CLASS,VECTOR).EQ.1.0) THEN
               CCOUNT=CCOUNT+1
C              DO UNTIL(VECTOR NUMBER "VECTOR" INCLUDED)
               DO 2 FETUR=FFEAT,LFEAT
                 PROTO(FETUR,CLASS)=PROTO(FETUR,CLASS)+
     1                              X(FETUR,VECTOR)
2              CONTINUE
C              END DO UNTIL
             END IF
C
3          CONTINUE
C          END DO UNTIL
C
C          DO UNTIL(MEAN VECTOR DIVIDED BY "CCOUNT")
           DO 4 FETUR=FFEAT,LFEAT
             PROTO(FETUR,CLASS)=PROTO(FETUR,CLASS)/CCOUNT
4          CONTINUE
C          END DO UNTIL
C
5        CONTINUE
C        END DO UNTIL
C
C   SET ITERATIONS TO ZERO INITIALLY
C
         ITERAT=0
C
C    * * * BEGIN ITERATIONS * * *
C
C        DO UNTIL(CLUSTER MEANS STABILIZE OR MAXIMUM
C                 ITERATIONS COMPLETED)
```

```
6       CONTINUE
C
C         DO UNTIL(ALL VECTORS ASSIGNED MEMBERSHIP
C                  VALUE=1.0 IN CLOSEST CLASS( MINIMUM
C                  DISTANCE TO PROTO) AND VALUE=0.0
C                  IN OTHER CLASSES)
          DO 10 VECTOR=1,VCOUNT
C
C           DO UNTIL(MINIMUM DISTANCE- CURRENT VECTOR
C                    TO ALL MEANS FOUND)
            DO 8 CLASS=1,KLASES
C
              XDIST=0.0
C
C             DO UNTIL(DISTANCE SQUARED COMPUTED FOR
C                      CURRENT VECTOR AND CLASS PROTO)
              DO 7 FETUR=FFEAT,LFEAT
                TEMP=X(FETUR,VECTOR)-PROTO(FETUR,CLASS)
                XDIST=XDIST+TEMP*TEMP
7             CONTINUE
C             END DO UNTIL
C
              IF(CLASS.EQ.1) THEN
                DISTMN=XDIST
                CLOSER=1
              ELSE IF(XDIST.LT.DISTMN) THEN
                DISTMN=XDIST
                CLOSER=CLASS
              END IF
C
8           CONTINUE
C           END DO UNTIL
C
C           DO UNTIL(ALL MEMBERSHIP FUNCTIONS OF
C                    CURRENT VECTOR ZEROED)
            DO 9 CLASS=1,KLASES
              MFUNCT(CLASS,VECTOR)=0.0
9           CONTINUE
C           END DO UNTIL
C
C   ASSIGN VECTOR TO CLASS NUMBER "CLOSER"
C
            MFUNCT(CLOSER,VECTOR)=1.0
C
10        CONTINUE
C         END DO UNTIL
C
C         DO UNTIL(NEW CLASS MEANS COMPUTED AND
C                  MAXIMUM UPDATE DIFFERENCE FOUND)
          DO 16 CLASS=1,KLASES
C
            CCOUNT=0
C           DO UNTIL(TEMPORARY MEAN VECTOR ZEROED)
            DO 11 FETUR=FFEAT,LFEAT
              UPDATE(FETUR)=0.0
11          CONTINUE
C           END DO UNTIL
C
C           DO UNTIL(CLASS MEAN FOR CURRENT CLASS SUMMED)
            DO 13 VECTOR=1,VCOUNT
C
              IF(MFUNCT(CLASS,VECTOR).EQ.1.0) THEN
                CCOUNT=CCOUNT+1
C               DO UNTIL(VECTOR "VECTOR" INCLUDED)
                DO 12 FETUR=FFEAT,LFEAT
                  UPDATE(FETUR)=UPDATE(FETUR)+
     1                          X(FETUR,VECTOR)
12              CONTINUE
C               END DO UNTIL
              END IF
C
13          CONTINUE
C           END DO UNTIL
```

```
C
C           DO UNTIL(MEAN VECTOR DIVIDED BY "CCOUNT")
            DO 14 FETUR=FFEAT,LFEAT
              UPDATE(FETUR)=UPDATE(FETUR)/CCOUNT
14          CONTINUE
C           END DO UNTIL
C
            DIST=0.0
C
C           DO UNTIL(DISTANCE SQUARED UPDATE MEAN TO
C                    PREVIOUS MEAN FOUND, AND NEW MEAN
C                    ASSIGNED AS THE UPDATE VECTOR FOUND)
            DO 15 FETUR=FFEAT,LFEAT
C
              TEMP=PROTO(FETUR,CLASS)-UPDATE(FETUR)
              DIST=DIST+TEMP*TEMP
              PROTO(FETUR,CLASS)=UPDATE(FETUR)
C
15          CONTINUE
C           END DO UNTIL
C
            IF(CLASS.EQ.1) THEN
              DIFMAX=DIST
            ELSE IF(DIST.GT.DIFMAX) THEN
              DIFMAX=DIST
            END IF
C
16        CONTINUE
C         END DO UNTIL
C
C  INCREMENT ITERATION COUNT
C
         ITERAT=ITERAT+1
C
        IF((DIFMAX.GT.EPSLON).AND.(ITERAT.LE.MAXITR))
     1                                                GO TO 6
C       END DO UNTIL
C
        RETURN
        END
```

```
$OPTIMIZE
C   A SUBROUTINE WHICH IMPLEMENTS A FUZZY VERSION
C   OF THE NEAREST NEIGHBOR ALGORITHM. THE RESULT
C   IS TO ASSIGN MEMBERSHIP FUNCTION VALUES TO THE
C   VECTOR TO BE CLASSIFIED INSTEAD OF ASSIGNING
C   THE VECTOR TO ONE OF THE CLASSES REPRESENTED
C   BY THE LABELLED DATA USED.
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: MAR 84
C
C   FILENAME: FUZNEARN.FTN
C
C   CALLING SEQUENCE(FROM A FORTRAN ROUTINE): CALL
C    FUZNN(FZFIER,KLASES,FFEAT,LFEAT,VCOUNT,K,TEST1)
C
C   INPUT VARIABLES(NOT CHANGED):
C
C    FZFIER - REAL VALUE OF THE WEIGHTING FACTOR
C             USED IN THE ALGORITHM
C
C    KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C    FFEAT - INTEGER WHICH SETS FIRST FEATURE IN
C            DATA VECTORS TO CONSIDER
C
C    LFEAT - INTEGER WHICH SETS LAST FEATURE IN
C            DATA VECTORS TO CONSIDER
C
C    VCOUNT - INTEGER COUNT OF NUMBER OF DATA
C             VECTOR SAMPLES
C
C    K - INTEGER COUNT OF NUMBER OF NEAREST NEIGHBORS
C        TO USE FOR MEMBERSHIP FUNCTION ASSIGNMENTS
C
C    TEST1 - INTEGER INDEX WHICH INDICATES WHICH OF
C            THE DATA VECTORS IS THE CURRENT TEST
C            SAMPLE TO BE ASSIGNED MEMBERSHIPS
C
C    X - REAL ARRAY 4 BY 242 WHICH HOLDS ALL DATA VECTORS,
C        PASSED IN LABELLED FORTRAN COMMOM "AREA1"
C
C    MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS
C             MEMBERSHIPS OF LABELLED SAMPLES USED
C             IN ASSIGNMENT OF "TEST1"'S MEMBERSHIPS,
C             PASSED IN LABELLED FORTRAN COMMON "AREA3"
C
C   OUTPUT VARIABLES:
C
C    NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C             MEMBERSHIP ASSIGNMENTS PRODUCED,
C             PASSED IN LABELLED FORTRAN COMMON "AREA5"
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                      PSEUDO - CODE SOLUTION
C
C   ENTER FUZNEARN
C     INITIALIZE LABELLED SAMPLE INDEX
C     INITIALIZE MATCH = NO
C     INITIALIZE NEAREST NEIGHBOR COUNTER
C     DO UNTIL( "K" NEAREST NEIGHBORS FOUND
C                        OR A MATCH FOUND)
C       COMPUTE DISTANCE FROM TEST VECTOR
C        TO CURRENT LABELLED SAMPLE
C       IF ( DISTANCE = 0 ) THEN
C         MATCH = YES
C       ELSE IF ( NEAREST NEIGHBOR  COUNTER <=
C                                          "K" ) THEN
C         INCLUDE CURRENT LABELLED SAMPLE
C          IN LIST OF NEAREST NEIGHBORS
C         INCREMENT NEAREST NEIGHBOR COUNTER
C       ELSE IF (DISTANCE LESS THAN THAT
```

```
C                OF ANY PREVIOUS NEAREST NEIGHBOR ) THEN
C          DELETE FARTHEST OF PREVIOUS NEAREST NEIGHBORS
C          INCLUDE NEW NEAREST NEIGHBOR IN LIST
C        END IF
C        INCREMENT LABELLED SAMPLE INDEX
C      END DO UNTIL
C      IF ( MATCH = NO ) THEN
C        ASSIGN MEMBERSHIPS OF MATCH LABELLED
C         SAMPLE TO TEST VECTOR
C      ELSE
C         ASSIGN MEMBERSHIPS AS A FUNCTION OF INVERSE
C          DISTANCES FROM NEAREST NEIGHBORS
C      END IF
C    RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       SUBROUTINE FUZNN(FZFIER,KLASES,FFEAT,LFEAT,
      1                 VCOUNT,K,TEST1)
C
       LOGICAL MATCH
       REAL MFUNCT(3,242),X(4,242),NMFUNC(3,242),DNEAR(10)
       INTEGER FFEAT,CLASS,NEAR(10),FETUR
       INTEGER TEST1,COUNTR,VTRAIN,VCOUNT
       COMMON /AREA3/MFUNCT /AREA1/X /AREA5/NMFUNC
C
C    COMPUTE POWER DEPENDENT ON "FUZZIFIER"
C
       POWER=1.0/(FZFIER-1.0)
C
           MATCH=.FALSE.
           DSTMAX=0.0
           VTRAIN=1
           COUNTR=1
C
C          DO UNTIL("K" NEAREST NEIGHBORS FOUND
C                   OR A MATCH OCCURS)
1          CONTINUE
C
             DIST=0.0
C
             IF(VTRAIN.NE.TEST1) THEN

C              DO UNTIL(DISTANCE COMPUTED)
               DO 3 FETUR=FFEAT,LFEAT
                 TEMP=X(FETUR,VTRAIN)-X(FETUR,TEST1)
                 DIST=DIST+TEMP*TEMP
3              CONTINUE
C              END DO UNTIL
C
               IF(DIST.EQ.0.0) THEN
                 MATCH=.TRUE.
                 MATNUM=VTRAIN
               ELSE IF(COUNTR.LE.K) THEN
                 DNEAR(COUNTR)=DIST
                 NEAR(COUNTR)=VTRAIN
                 IF(DNEAR(COUNTR).GT.DSTMAX) THEN
                   DSTMAX=DNEAR(COUNTR)
                   MAXNER=COUNTR
                 END IF
                 COUNTR=COUNTR+1
               ELSE IF(DIST.LT.DSTMAX) THEN
                 DSTMAX=DIST
                 DNEAR(MAXNER)=DIST
                 NEAR(MAXNER)=VTRAIN
C                DO UNTIL(NEW MAXIMUM DISTANCE OF
C                         K-NEAREST NEIGHBORS FOUND)
                 DO 4 INDEX=1,K
                   IF(DNEAR(INDEX).GT.DSTMAX) THEN
                     DSTMAX=DNEAR(INDEX)
                     MAXNER=INDEX
                   END IF
4                CONTINUE
C                END DO UNTIL
```

```
            END IF
C
          END IF
C
          VTRAIN=VTRAIN+1
C
        IF((.NOT.MATCH).AND.(VTRAIN.LE.VCOUNT))
     1GO TO 1
C       END DO UNTIL
C
        IF(.NOT.MATCH) THEN
C
C         DO UNTIL(MEMBERSHIP FUNCTIONS ASSIGNED
C                  TO VECTOR NUMBER "VTEST")
          DO 6 CLASS=1,KLASES
C
            NMFUNC(CLASS,TEST1)=0.0
            SUM=0.0
C           DO UNTIL(MEMBERSHIP FUNCTION COMPUTED
C                    FOR CLASS NUMBER "CLASS")
            DO 5 INDEX=1,K
              WDIST= 1.0/DNEAR(INDEX)**POWER
              SUM=SUM+WDIST
              NMFUNC(CLASS,TEST1)=NMFUNC(CLASS,TEST1)+
     1                 MFUNCT(CLASS,NEAR(INDEX))*WDIST
5           CONTINUE
C           END DO UNTIL
C
            NMFUNC(CLASS,TEST1)=NMFUNC(CLASS,TEST1)/SUM
C
6         CONTINUE
C         END DO UNTIL
C
        ELSE
C
C         DO UNTIL(MEMBERSHIP FUNCTIONS ASSIGNED)
          DO 7 CLASS=1,KLASES
            NMFUNC(CLASS,TEST1)=MFUNCT(CLASS,MATNUM)
7         CONTINUE
C         END DO UNTIL
C
        END IF
C
       RETURN
       END
```

```
$OPTIMIZE
C   A SUBROUTINE WHICH IMPLEMENTS A K-NEAREST
C   NEIGHBOR ALGORITHM. VECTORS ARE ASSIGNED
C   TO A CLASS VIA MEMBERSHIP FUNCTION ASSIGNMENT
C   WITH POSSIBLE VALUES=>0,1<
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: MAR 84
C
C   FILENAME: CRISPNN.FTN
C
C   CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C    CALL CRSPNN(KLASES,FFEAT,LFEAT,K,VCOUNT,TEST1)
C
C   INPUT VARIABLES(NOT CHANGED):
C
C    KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C    FFEAT - INTEGER WHICH SETS FIRST FEATURE IN
C            DATA VECTORS TO CONSIDER
C
C    LFEAT - INTEGER WHICH SETS LAST FEATURE IN
C            DATA VECTORS TO CONSIDER
C
C    K - INTEGER COUNT OF NUMBER OF NEAREST
C        NEIGHBORS TO USE FOR CLASSIFICATION
C
C    VCOUNT - INTEGER COUNT OF NUMBER OF
C             DATA VECTORS IN DATA SET
C
C    TEST1 - INTEGER INDEX OF SAMPLE WHICH
C            IS BEING CLASSIFIED
C
C    X - REAL ARRAY 4 BY 242 WHICH HOLDS ALL DATA VECTORS,
C        PASSED IN LABELLED FORTRAN COMMON "AREA1"
C
C    MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS MEMBERSHIPS
C             LABELLED SAMPLES USED IN CLASSIFICATION OF
C             "TEST1", PASSED IN LABELLED FORTRAN
C             COMMON "AREA3"
C
C   OUTPUT VARIABLES:
C
C    NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C             MEMBERSHIP ASSIGNMENTS OF SAMPLE
C             "TEST1", PASSED IN LABELLED
C             FORTRAN COMMON "AREA3"
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                        PSEUDO - CODE SOLUTION
C
C   ENTER CRSPNN
C     INITIALIZE LABELLED SAMPLE INDEX
C     INITIALIZE MATCH = NO
C     INITIALIZE NEAREST NEIGHBOR COUNTER TO 0
C     DO UNTIL("K" NEAREST NEIGHBORS FOUND
C              OR A MATCH FOUND)
C       COMPUTE DISTANCE FROM TEST VECTOR
C        TO CURRENT LABELLED SAMPLE
C       IF(DISTANCE = 0 ) THEN
C         MATCH = YES
C       ELSE IF ( NEAREST NEIGHBOR COUNTER <= "K") THEN
C         INCLUDE CURRENT LABELLED SAMPLE IN
C          LIST OF K NEAREST NEIGHBORS
C         INCREMENT NEAREST NEIGHBOR COUNTER
C       ELSE IF(DISTANCE LESS THAN THAT OF ANY
C                        PREVIOUS NEAREST NEIGHBOR) THEN
C         DELETE FARTHEST OF PREVIOUS NEAREST NEIGHBORS
C         INCLUDE NEW NEAREST NEIGHBOR IN LIST
C       INCREMENT LABELLED SAMPLE INDEX
C     END DO UNTIL
C     IF (MATCH = NO ) THEN
C       COUNT NUMBER OF NEAREST NEIGHBORS
```

```
C        FROM EACH CLASS
C       IF(A TIE FOR MAXIMUM NUMBER BETWEEN CLASSES) THEN
C         COMPUTE SUM OF DISTANCES FROM TEST VECTOR
C          TO ALL NEIGHBORS IN EACH TYING CLASS
C         IF( TIE IN SUMS COMPUTED) THEN
C           ASSIGN TEST VECTOR TO LAST CLASS WHICH TIED
C         ELSE
C           ASSIGN TEST VECTOR TO CLASS OF MINIMUM SUM
C         END IF
C       ELSE
C         ASSIGN TEST VECTOR TO CLASS OF
C          MAXIMUM NUMBER OF NEIGHBORS
C       END IF
C     ELSE
C       ASSIGN TEST VECTOR TO CLASS OF MATCH
C     END IF
C   RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE CRSPNN(KLASES,FFEAT,LFEAT,K,VCOUNT,TEST1)
C
      LOGICAL MATCH
      REAL MFUNCT(3,242),X(4,242),NMFUNC(3,242)
      REAL DNEAR(10),DSUM(10)
      INTEGER FFEAT,CLASS,VECTOR,NEAR(10),FEATUR
      INTEGER VCOUNT,NEARCL(10),END(3),VTRAIN
      INTEGER COUNTR,COUNT(3),MAXNUM(10),TEST1,START(3)
      COMMON /AREA3/MFUNCT /AREA1/X /AREA5/NMFUNC
      COMMON /AREA6/ START,END
C
          MATCH=.FALSE.
          DSTMAX=0.0
          VTRAIN=1
          COUNTR=1
C
C         DO UNTIL("K" NEAREST NEIGHBORS FOUND
C                  OR A MATCH OCCURS)
1         CONTINUE
C
            IF(VTRAIN.NE.TEST1) THEN
C
              DIST=0.0
C
C             DO UNTIL(DISTANCE COMPUTED)
              DO 3 FEATUR=FFEAT,LFEAT
                TEMP=X(FEATUR,VTRAIN)-X(FEATUR,TEST1)
                DIST=DIST+TEMP*TEMP
3             CONTINUE
C             END DO UNTIL
C
              IF(DIST.EQ.0.0) THEN
                MATCH=.TRUE.
                MATNUM=VTRAIN
              ELSE IF(COUNTR.LE.K) THEN
                DNEAR(COUNTR)=DIST
                NEAR(COUNTR)=VTRAIN
                IF(DNEAR(COUNTR).GT.DSTMAX) THEN
                  DSTMAX=DNEAR(COUNTR)
                  MAXNER=COUNTR
                END IF
                COUNTR=COUNTR+1
              ELSE IF(DIST.LT.DSTMAX) THEN
                DSTMAX=DIST
                DNEAR(MAXNER)=DIST
                NEAR(MAXNER)=VTRAIN
C               DO UNTIL(NEW MAXIMUM DISTANCE OF K
C                        NEAREST NEIGHBORS FOUND)
                DO 4 INDEX=1,K
                  IF(DNEAR(INDEX).GT.DSTMAX) THEN
                    DSTMAX=DNEAR(INDEX)
                    MAXNER=INDEX
```

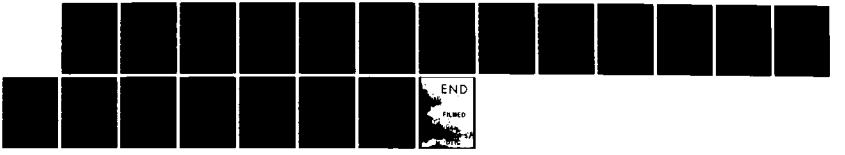
```
                END IF
4               CONTINUE
C               END DO UNTIL
              END IF
C
            END IF
            VTRAIN=VTRAIN+1
C
          IF((.NOT.MATCH).AND.(VTRAIN.LE.VCOUNT))
     1                                    GO TO 1
C         END DO UNTIL
C
          IF(.NOT.MATCH) THEN
C
C           DO UNTIL(COUNTS OF NEAREST NEIGHBORS
C                    CLASS'S AND CURRENT VECTOR'S
C                    MEMBERSHIP'S ZEROED)
            DO 5 CLASS=1,KLASES
              COUNT(CLASS)=0
              NMFUNC(CLASS,TEST1)=0.0
5           CONTINUE
C           END DO UNTIL
C
C           DO UNTIL(CLASS NUMBER OF K-NEAREST
C                    NEIGHBORS, AND COUNT OF
C                    NEAREST NEIGHBORS IN A CLASS FOUND)
            DO 7 INDEX=1,K
C
C             DO UNTIL("INDEX" NEIGHBOR'S
C                      CLASS FOUND)
              DO 6 CLASS=1,KLASES
                IF((NEAR(INDEX).GE.START(CLASS)).AND.
     1             (NEAR(INDEX).LE.END(CLASS))) THEN
                  COUNT(CLASS)=COUNT(CLASS)+1
                  NEARCL(INDEX)=CLASS
                END IF
6             CONTINUE
C             END DO UNTIL
C
7           CONTINUE
C           END DO UNTIL
C
C           DO UNTIL(COUNTS SEARCHED FOR MAXIMUM(S) )
            DO 8 CLASS=1,KLASES
              IF(CLASS.EQ.1) THEN
                MAX=COUNT(CLASS)
                MAXCNT=1
                MAXNUM(MAXCNT)=CLASS
              ELSE IF(COUNT(CLASS).GT.MAX) THEN
                MAX=COUNT(CLASS)
                MAXNUM(MAXCNT)=CLASS
              ELSE IF(COUNT(CLASS).EQ.MAX) THEN
                MAXCNT=MAXCNT+1
                MAXNUM(MAXCNT)=CLASS
              END IF
8           CONTINUE
C           END DO UNTIL
C
            IF(MAXCNT.EQ.1) THEN
              NMFUNC(MAXNUM(MAXCNT),TEST1)=1.0
            ELSE
C             DO UNTIL(SUM OF DISTANCES OF NEIGHBORS
C                      IN EACH CLASS WHICH TIED
C                      FOR A MAJORITY COMPUTED)
              DO 10 INDEX=1,MAXCNT
                DSUM(INDEX)=0.0
                DO 9 NEIBOR=1,K
                  IF(NEARCL(NEIBOR).EQ.MAXNUM(INDEX)) THEN
                    DSUM(INDEX)=DSUM(INDEX)+DNEAR(NEIBOR)
                  END IF
9               CONTINUE
10            CONTINUE
C             END DO UNTIL
```

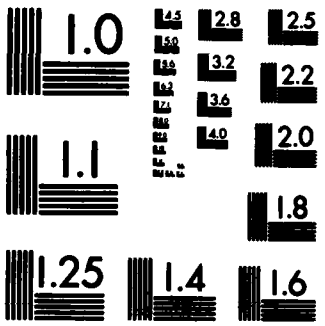
```
C              DO UNTIL(MAX OF SUMS OF DISTANCES
C                       COMPUTED ABOVE FOUND)
               DO 11 INDEX=1,MAXCNT
                 IF(INDEX.EQ.1) THEN
                   DMIN=DSUM(INDEX)
                   MIN=INDEX
                 ELSE IF(DSUM(INDEX).LE.DMIN) THEN
                   DMIN=DSUM(INDEX)
                   MIN=INDEX
                 END IF
11             CONTINUE
C              END DO UNTIL
C
C  ASSIGN VECTOR TO THE CLASS WITH MINIMUM
C  TOTAL DISTANCES(TIE GOES TO LAST MINIMUM FOUND)
C
               NMFUNC(NEARCL(MIN),TEST1)=1.0
             END IF
C
           ELSE
C
C            DO UNTIL(MEMBERSHIP FUNCTIONS ASSIGNED)
             DO 12 CLASS=1,KLASES
               NMFUNC(CLASS,TEST1)=MFUNCT(CLASS,MATNUM)
12           CONTINUE
C            END DO UNTIL
C
           END IF
C
      RETURN
      END
```

```
$OPTIMIZE
C   A ROUTINE WHICH IMPLEMENTS A FUZZY VERSION
C   OF THE NEAREST PROTOTYPE ALGORITHM.  THE
C   RESULT IS TO ASSIGN MEMBERSHIP FUNCTION
C   VALUES TO THE VECTOR TO BE CLASSIFIED
C   INSTEAD OF ASSIGNING THE VECTOR TO ONE
C   OF THE CLASSES REPRESENTED BY THE PROTOTYPE
C   DATA USED.
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: MAR 84
C
C   FILENAME: FUZPROTO.FTN
C
C   CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C    CALL FPROTO(FZFIER,KLASES,FFEAT,LFEAT,TEST1)
C
C   INPUT VARIABLES(NOT CHANGED):
C
C    FZFIER - REAL WEIGHTING FACTOR USED IN
C             THE MEMBERSHIP ASSIGNMENTS
C
C    KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C    FFEAT - INTEGER WHICH SETS THE FIRST
C            FEATURE IN THE DATA SET TO CONSIDER
C
C    LFEAT - INTEGER WHICH SETS THE LAST FEATURE
C            IN THE DATA SET TO CONSIDER
C
C    TEST1 - INTEGER INDEX WHICH POINTS TO VECTOR
C            IN DATA TO ASSIGN MEMBERSHIPS
C
C    X - REAL ARRAY 4 BY 242 WHICH HOLDS SAMPLES,
C        PASSED IN LABELLED FORTRAN COMMON "AREA1"
C
C    PROTO-REAL ARRAY 4 BY 3 WHICH HOLDS CLASS
C          PROTOTYPES, PASSED IN LABELLED
C          FORTRAN COMMON "AREA2"
C
C   OUTPUT VARIABLES:
C
C    NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C             MEMBERSHIP FUNCTIONS ASSIGNED, PASSED
C             IN LABELLED FORTRAN COMMON "AREA5"
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          PSEUDO - CODE SOLUTION
C
C   ENTER FPROTO
C     SET CLASS INDEX=1
C     INITIALIZE MATCH = NO
C     DO UNTIL(DISTANCE FROM TEST SAMPLE TO ALL
C              PROTOTYPES COMPUTED OR A MATCH FOUND)
C       COMPUTE DISTANCE FROM CURRENT CLASS
C        PROTOTYPE TO TEST SAMPLE
C       IF (DISTANCE = 0 ) THEN
C         MATCH = YES
C       END IF
C       INCREMENT CLASS INDEX
C     END DO UNTIL
C     IF ( MATCH ) THEN
C       ASSIGN TEST SAMPLE MEMBERSHIP OF
C        PROTOTYPE WHICH MATCHED
C     ELSE
C       ASSIGN MEMBERSHIPS AS A FUNCTION
C        OF INVERSE DISTANCES COMPUTED
C     END IF
C   RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

AD-A145 571 AN INVESTIGATION INTO FUZZY CLUSTERING AND CLASSIFICATION(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M R GRAY JUL 84 2/2

UNCLASSIFIED AFIT/CI/NR-84-47T F/G 12/1 NL





```
      SUBROUTINE FPROTO(FZFIER,KLASES,FFEAT,LFEAT,TEST1)
C
      LOGICAL MATCH
      REAL PROTO(4,3),NMFUNC(3,242),PDIST(3),X(4,242)
      INTEGER FFEAT,CLASS,VECTOR,FETUR,TEST1
      COMMON /AREA2/PROTO /AREA1/X /AREA5/NMFUNC
C
C  COMPUTE POWER WHICH IS A FUNCTION OF THE "FUZZIFIER"
C
      POWER=1.0/(FZFIER-1.0)
C
          DSUM=0.0
          CLASS=1
C
C         DO UNTIL(DISTANCE CURRENT VECTOR TO EACH
C                  PROTOTYPE, OR A MATCH FOUND)
1         CONTINUE
C
            MATCH=.FALSE.
            PDIST(CLASS)=0.0
C
C           DO UNTIL(DISTANCE COMPUTED)
            DO 2 FETUR=FFEAT,LFEAT
              TEMP=PROTO(FETUR,CLASS)-X(FETUR,TEST1)
              PDIST(CLASS)=PDIST(CLASS)+TEMP*TEMP
2           CONTINUE
C           END DO UNTIL
C
            IF(PDIST(CLASS).EQ.0.0) THEN
              MATCH=.TRUE.
              MNUM=CLASS
            ELSE
              PDIST(CLASS)=1.0/PDIST(CLASS)**POWER
              DSUM=DSUM+PDIST(CLASS)
            END IF
C
            CLASS=CLASS+1
C
          IF((.NOT.MATCH).AND.(CLASS.LE.KLASES))
     1                                    GO TO 1
C         END DO UNTIL
C
          IF(MATCH) THEN
C           DO UNTIL(MEMBERSHIP FUNCTIONS OF
C                    CURRENT VECTOR ZEROED)
            DO 3 CLASS=1,KLASES
               NMFUNC(CLASS,TEST1)=0.0
3           CONTINUE
C           END DO UNTIL
C
C  ASSIGN THE VECTOR MEMBERSHIP IN THE
C  CLASS FOR WHICH A MATCH OCCURRED
C
            NMFUNC(MNUM,TEST1)=1.0
C
          ELSE
C           DO UNTIL(MEMBERSHIP FUNCTIONS
C                    ASSIGNED FOR TEST VECTOR)
            DO 4 CLASS=1,KLASES
              NMFUNC(CLASS,TEST1)=PDIST(CLASS)/DSUM
4           CONTINUE
C           END DO UNTIL
          END IF
C
       RETURN
       END
```

```
$OPTIMIZE
C  A SUBROUTINE WHICH IMPLEMENTS A CRISP VERSION
C  OF THE NEAREST PROTOTYPE ALGORITHM.  THE
C  RESULT IS TO ASSIGN MEMBERSHIP FUNCTION
C  VALUES TO THE VECTOR TO BE CLASSIFIED
C  INSTEAD OF ASSIGNING THE VECTOR TO ONE OF
C  THE CLASSES REPRESENTED BY THE PROTOTYPES
C
C  WRITTEN BY: MICHAEL R. GRAY
C
C  COMPLETED: APR 84
C
C  FILENAME: CRISPNP.FTN
C
C  CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C          CALL CRSNP(KLASES,FFEAT,LFEAT,TEST1)
C
C  INPUT VARIABLES(NOT CHANGED):
C
C   KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C   FFEAT - INTEGER WHICH SETS THE FIRST FEATURE
C           IN DATA SET TO CONSIDER
C
C   LFEAT - INTEGER WHICH SETS THE LAST FEATURE
C           IN DATA SET TO CONSIDER
C
C   TEST1 - INTEGER WHICH HOLDS THE INDEX
C           OF THE CURRENT TEST SAMPLE
C
C   X - REAL ARRAY 4 BY 242  WHICH HOLDS ALL DATA
C       VECTORS, PASSED IN LABELLED FORTRAN COMMON "AREA1"
C
C   PROTO - REAL ARRAY 4 BY 3 WHICH HOLDS THE CLASS
C           PROTOTYPES USED IN MEMBERSHIP FUNCTION
C           ASSIGNMENT, PASSED IN LABELLLED FORTRAN
C           COMMON "AREA2"
C
C  OUTPUT VARIABLES:
C
C   NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS
C            THE RESULTING MEMBERSHIP FUNCTION
C            ASSIGNMENTS FOR THE CURRENT TEST VECTOR
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                      PSEUDO - CODE SOLUTION
C
C  ENTER CRSPNP
C    INITIALIZE MATCH = NO
C    INITIALIZE CLASS INDEX=1
C    DO UNTIL(DISTANCES FROM TEST SAMPLE TO EACH
C             PROTOTYPE COMPUTED OR A MATCH FOUND)
C      COMPUTE DISTANCE FROM TEST SAMPLE
C       TO CURRENT CLASS PROTOTYPE
C      IF(DISTANCE = 0 ) THEN
C        MATCH = YES
C      END IF
C      INCREMENT CLASS INDEX
C    END DO UNTIL
C    IF(MATCH = YES) THEN
C      ASSIGN TEST VECTOR MEMBERSHIPS
C       OF MATCHING PROTOTYPE
C    ELSE
C      DETERMINE CLASS OF CLOSEST PROTOTYPE
C      IF ( A TIE EXISTS) THEN
C        ASSIGN TEST SAMPLE TO CLASS OF LAST
C         CLOSE PROTOTYPE DETECTED
C      ELSE
C        ASSIGN TEST SAMPLE TO CLASS OF
C         CLOSEST PROTOTYPE
C      END IF
C    END IF
C  RETURN
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE CRSPNP(KLASES,FFEAT,LFEAT,TEST1)
C
      LOGICAL MATCH
      REAL PROTO(4,3),NMFUNC(3,242),PDIST(3),X(4,242)
      INTEGER FFEAT,CLASS,VECTOR,FETUR,TEST1
      COMMON /AREA2/PROTO /AREA1/X /AREA5/NMFUNC
C
          DSUM=0.0
          CLASS=1
          MATCH=.FALSE.
C
C         DO UNTIL(DISTANCE CURRENT VECTOR TO EACH
C                  PROTOTYPE COMPUTED, OR A MATCH FOUND)
1         CONTINUE
C
            PDIST(CLASS)=0.0
C
C           DO UNTIL(DISTANCE COMPUTED)
            DO 2 FETUR=FFEAT,LFEAT
              TEMP=PROTO(FETUR,CLASS)-X(FETUR,TEST1)
              PDIST(CLASS)=PDIST(CLASS)+TEMP*TEMP
2           CONTINUE
C           END DO UNTIL
C
C
            IF(PDIST(CLASS).EQ.0.0) THEN
              MATCH=.TRUE.
              MNUM=CLASS
            END IF
C
            CLASS=CLASS+1
C
          IF((.NOT.MATCH).AND.(CLASS.LE.KLASES))
     1                                    GO TO 1
C         END DO UNTIL
C
C         DO UNTIL(MEMBERSHIP FUNCTIONS OF
C                  CURRENT VECTOR ZEROED)
          DO 3 CLASS=1,KLASES
            NMFUNC(CLASS,TEST1)=0.0
3         CONTINUE
C         END DO UNTIL
C
          IF(MATCH) THEN
C
C   ASSIGN THE VECTOR MEMBERSHIP IN THE CLASS
C   FOR WHICH A MATCH OCCURRED
C
            NMFUNC(MNUM,TEST1)=1.0
C
          ELSE
C           DO UNTIL(CLOSEST OF THE PROTOTYPES FOUND)
            DO 4 CLASS=1,KLASES
              IF(CLASS.EQ.1) THEN
                CLOSE=PDIST(CLASS)
                NUMBER=CLASS
              ELSE IF(PDIST(CLASS).LE.CLOSE) THEN
                CLOSE=PDIST(CLASS)
                NUMBER=CLASS
              END IF
4           CONTINUE
C           END DO UNTIL
C
C   ASSIGN VECTOR TO CLASS OF CLOSEST PROTOTYPE
C    (IF A TIE CLOSEST=LAST CLOSEST FOUND)
C
            NMFUNC(NUMBER,TEST1)=1.0
C
          END IF
C
       RETURN
```

END

```
$OPTIMIZE
C   A SUBROUTINE WHICH ASSIGNS MEMBERSHIP
C   FUNCTION VALUES BASED ON THE NEAREST
C   NEIGHBOR "FUZZIFYING" RULE,
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: JUNE 84
C
C   FILENAME: MGFZFYNN.FTN
C
C   CALLING SEQUENCE(FROM A FORTRAN ROUTINE): CALL
C    FZFYNN(FZFIER,KLASES,VCOUNT,FFEAT,LFEAT,K,TEST1)
C
C   INPUT VARIABLES(NOT CHANGED):
C
C    FZFIER - REAL VALUE OF THE WEIGHTING FACTOR
C
C    KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C    VCOUNT - INTEGER COUNT OF NUMBER OF SAMPLE VECTORS
C
C    FFEAT - INTEGER WHICH SET FIRST FEATURE
C            IN SAMPLE VECTORS TO CONSIDER
C
C    LFEAT - INTEGER WHICH SET LAST FEATURE IN
C            IN SAMPLE VECTOR TO CONSIDER
C
C    K - INTEGER COUNT OF NUMBER OF NEIGHBORS TO
C        USE FOR MEMBERSHIP FUNCTION INITIALIZATION
C
C    TEST1 - INTEGER INDEX OF TEST SAMPLE
C
C    X - REAL ARRAY 4 BY 242 WHICH HOLDS ALL SAMPLE VECTORS,
C        PASSED IN LABELLED FORTRAN COMMON "AREA1"
C
C    START - INTEGER ARRAY OF DIMENSION 3 WHICH HOLDS THE
C            STARTING INDICES OF EACH CLASS OF SAMPLES,
C            PASSED IN LABELLED FORTRAN COMMON "AREA6"
C
C    END - INTEGER ARRAY OF DIMENSION 3 WHICH HOLDS
C          THE ENDING INDICES OF EACH CLASS OF SAMPLES,
C          PASSED IN LABELLED FORTRAN COMMON "AREA6"
C
C   OUTPUT VARIABLES:
C
C    MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS
C             THE MEMBERSHIP FUNCTION ASSIGNMENTS,
C             PASSED IN LABELLED FORTRAN COMMON "AREA3"
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          PSEUDO - CODE SOLUTION
C
C   ENTER FZFYNN
C    INITIALIZE VECTOR INDEX = 1
C    DO UNTIL ( EACH LABELLED SAMPLE USED TO CLASSIFY
C               "TEST1" ASSIGNED MEMBERSHIPS)
C      IF(VECTOR NOT EQUAL TO "TEST1") THEN
C        FIND K NEAREST NEIGHBORS IN LABELLED
C         SAMPLE SET TO CURRENT VECTOR
C        DETERMINE COUNT OF NEAREST NEIGHBORS
C         TO USE FOR INITIALIZATION
C        INITIALIZE CLASS INDEX = 1
C        DO UNTIL (CURRENT VECTOR ASSIGNED
C                  MEMBERSHIP IN ALL CLASSES)
C          IF(CLASS INDEX = KNOWN CLASS
C                           OF CURRENT VECTOR) THEN
C            ASSIGN MEMBERSHIP = 0.51 + (COUNT OF
C             "K" NEAREST NEIGHBORS FOUND/"K") * 0.49
C          ELSE
C            ASSIGN MEMBERSHIP = (COUNT OF "K"
C             NEAREST NEIGHBORS FOUND/"K")*0.49
C          END IF
```

```
C          INCREMENT CLASS INDEX
C        END DO UNTIL
C      END IF
C      INCREMENT VECTOR INDEX
C    END DO UNTIL
C  RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       SUBROUTINE FZFYNN(FZFIER,KLASES,VCOUNT,FFEAT,
      1                  LFEAT,K,TEST1)
C
       REAL MFUNCT(3,242),X(4,242),DNEAR(10)
       INTEGER FFEAT,COUNTR,VECTOR,NEAR(10)
       INTEGER START(3),END(3),VCOUNT,TEST1,VTRAIN
       INTEGER FETUR,CLASS,COUNT(3),VCLASS
       COMMON /AREA3/MFUNCT /AREA1/X /AREA6/START,END
C
C
C      DO UNTIL(ALL DATA ASSIGNED MEMBERSHIPS)
       DO 40 VTRAIN=1,VCOUNT
C
         IF(VTRAIN.NE.TEST1) THEN
C
           COUNTR=1
           DSTMAX=0.0
           MAXNER=1
C
C          DO UNTIL(K NEAREST NEIGHBORS FOUND)
           DO 5 VECTOR=1,VCOUNT
C
             IF((VECTOR.NE.VTRAIN).AND.
      1(VECTOR.NE.TEST1)) THEN
C
               DIST=0.0
C
C              DO UNTIL(DISTANCE COMPUTED)
               DO 3 FETUR=FFEAT,LFEAT
                 TEMP=X(FETUR,VTRAIN)-X(FETUR,VECTOR)
                 DIST=DIST+TEMP*TEMP
3              CONTINUE
C              END DO UNTIL
C
               IF(COUNTR.LE.K) THEN
                 DNEAR(COUNTR)=DIST
                 NEAR(COUNTR)=VECTOR
                 IF(DNEAR(COUNTR).GT.DSTMAX) THEN
                   DSTMAX=DNEAR(COUNTR)
                   MAXNER=COUNTR
                 END IF
                 COUNTR=COUNTR+1
               ELSE IF(DIST.LT.DNEAR(MAXNER)) THEN
                 DNEAR(MAXNER)=DIST
                 NEAR(MAXNER)=VECTOR
C                DO UNTIL(NEW MAXIMUM DISTANCE OF
C                         K NEAREST NEIGHBORS FOUND)
                 DO 4 INDEX=1,K
                   IF(DNEAR(INDEX).GT.DNEAR(MAXNER)) THEN
                     MAXNER=INDEX
                   END IF
4                CONTINUE
C                END DO UNTIL
               END IF
C
             END IF
C
5          CONTINUE
C          END DO UNTIL
C
C          DO UNTIL(CLASS OF NEAREST NEIGHBORS
C                   AND "VTRAIN" DETERMINED)
           DO 10 CLASS=1,KLASES
             COUNT(CLASS)=0
             DO 7 INDEX=1,K
```

```
                IF((NEAR(INDEX).GE.START(CLASS)).AND.
     1              (NEAR(INDEX).LE.END(CLASS))) THEN
                  COUNT(CLASS)=COUNT(CLASS)+1
                END IF
7             CONTINUE
              IF((VTRAIN.GE.START(CLASS)).AND.
     1              (VTRAIN.LE.END(CLASS))) THEN
                VCLASS=CLASS
              END IF
10          CONTINUE
C           END DO UNTIL
C
C           DO UNTIL(MEMBERSHIP FOR VECTOR
C                    NUMBER "VTRAIN" ASSIGNED)
            DO 15 CLASS=1,KLASES
              IF(CLASS.EQ.VCLASS) THEN
                MFUNCT(CLASS,VTRAIN)=0.51+
     1                (COUNT(CLASS)*0.49)/K
              ELSE
                MFUNCT(CLASS,VTRAIN)=(COUNT(CLASS)*0.49)/K
              END IF
15          CONTINUE
C           END DO UNTIL
C
          END IF
C
40      CONTINUE
C       END DO UNTIL
C
        RETURN
        END
```

```
$OPTIMIZE
C
C   A SUBROUTINE WHICH INITIALIZES THE MEMBERSHIP
C   ARRAY USED IN A FUZZY CLASSIFICATION ALGORITHM.
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: APR 84
C
C   CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C    CALL INITMF(VCOUNT,FFEAT,LFEAT,KLASES,HIGH,LOW)
C
C   INPUT VARIABLES(NOT CHANGED):
C
C    VCOUNT - THE NUMBER OF VECTORS IN VECTOR, AN INTEGER.
C
C    FFEAT - THE FIRST FEATURE TO BE CONSIDERED
C            IN THE DATA VECTORS, AN INTEGER.
C
C    LFEAT - THE LAST FEATURE TO BE CONSIDERED
C            IN THE DATA VECTORS, AN INTEGER.
C
C    HIGH - THE REAL VALUE TO USE FOR "HIGH" MEMBERSHIP.
C
C    LOW - THE REAL VALUE TO USE FOR "LOW" MEMBERSHIP.
C
C    X - A REAL ARRAY OF DIMENSION (4,242), DATA SET
C        TO BE CLASSIFIED OR CLUSTERED, PASSED IN
C        LABELLED FORTRAN COMMON "AREA1"
C
C   OUTPUT VARIABLES:
C
C    MFUNCT - A REAL ARRAY OF DIMENSION (3,242),CONTAINS
C             "KLASES" CLASSES, "VCOUNT" VECTORS,
C             PASSED IN LABELLED FORTRAN COMMON "AREA3"
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                  PSEUDO - CODE SOLUTION
C
C   ENTER INITMF
C    COMPUTE MEAN OF ENTIRE DATA SET
C    FIND SAMPLE FARTHEST FROM SAMPLE
C     MEAN AND ASSIGN IT AS "FAR1"
C    IF( KLASES >= 3 ) THEN
C      FIND SAMPLE FARTHEST FROM "FAR1"
C       AND ASSIGN IT AS "FAR2"
C    END IF
C    ASSIGN ALL SAMPLES WITH "HIGH" MEMBERSHIP IN
C     CLASS 2 AND "LOW" MEMBERSHIP IN ALL OTHER CLASSES
C    IF( CLASSES < = 2 ) THEN
C      REASSIGN SAMPLE "FAR1" WITH "HIGH" MEMBERSHIP IN
C       CLASS 1 AND "LOW" MEMBERSHIP IN ALL OTHER CLASSES
C    ELSE IF (CLASSES >= 3) THEN
C      REASSIGN SAMPLE "FAR1" WITH "HIGH" MEMBERSHIP IN
C       CLASS 3 AND "LOW" MEMBERSHIP IN ALL OTHER CLASSES
C      REASSIGN SAMPLE "FAR1" WITH "HIGH" MEMBERSHIP IN
C       CLASS 1 AND "LOW" MEMBERSHIP IN ALL OTHER CLASSES
C    END IF
C   RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE INITMF(VCOUNT,FFEAT,LFEAT,KLASES,HIGH,LOW)
C
      INTEGER FFEAT,FETUR,VECTOR,VCOUNT,FAR1,FAR2
      REAL MFUNCT(3,242),X(4,242),SMEAN(4),LOW
      COMMON /AREA3/MFUNCT /AREA1/X
C
C     DO UNTIL(SAMPLE MEAN VECTOR SET TO ZERO)
      DO 1 FETUR=FFEAT,LFEAT
       SMEAN(FETUR)=0.0
1     CONTINUE
C     END DO UNTIL
```

```
C
C       DO UNTIL(ALL VECTOR IN DATA SET SUMMED)
        DO 3 VECTOR=1,VCOUNT
C        DO UNTIL(VECTOR NUMBER "VECTOR" INCLUDED IN SUM)
         DO 2 FETUR=FFEAT,LFEAT
          SMEAN(FETUR)=SMEAN(FETUR)+X(FETUR,VECTOR)
2        CONTINUE
C        END DO UNTIL
3       CONTINUE
C       END DO UNTIL
C
C       DO UNTIL(ALL COMPONENTS OF SAMPLE MEAN VECTOR
C                DIVIDED BY COUNT OF VECTORS IN DATA SET)
        DO 4 FETUR=FFEAT,LFEAT
         SMEAN(FETUR)=SMEAN(FETUR)/VCOUNT
4       CONTINUE
C       END DO UNTIL
C
        DMAX=0.0
C
C       DO UNTIL(VECTOR FARTHEST FROM SAMPLE MEAN FOUND)
        DO 6 VECTOR=1,VCOUNT
C
          DIST1=0.0
C
C         DO UNTIL(DISTANCE SQUARED FROM CURRENT
C                  VECTOR TO SAMPLE MEAN FOUND)
          DO 5 FETUR=FFEAT,LFEAT
            TEMP=X(FETUR,VECTOR)-SMEAN(FETUR)
            DIST1=DIST1+TEMP*TEMP
5         CONTINUE
C         END DO UNTIL
C
          IF(DIST1.GT.DMAX) THEN
            DMAX=DIST1
            FAR1=VECTOR
          END IF
C
6       CONTINUE
C       END DO UNTIL
C
        IF(KLASES.GE.3) THEN
          DMAX=0.0
C
C         DO UNTIL(VECTOR FARTHEST FROM THE
C                  ONE FOUND ABOVE LOCATED)
          DO 8 VECTOR=1,VCOUNT
C
            DIST1=0.0
C
C           DO UNTIL(DISTANCE SQUARED FROM CURRENT
C                    VECTOR TO VECOTR "FAR1" FOUND)
            DO 7 FETUR=FFEAT,LFEAT
            TEMP=X(FETUR,VECTOR)-X(FETUR,FAR1)
            DIST1=DIST1+TEMP*TEMP
7           CONTINUE
C           END DO UNTIL
C
            IF(DIST1.GT.DMAX) THEN
              DMAX=DIST1
              FAR2=VECTOR
            END IF
C
8         CONTINUE
C         END DO UNTIL
        END IF
C
C  ASSIGN MEMBERSHIP FUNCTIONS AS FOLLOWS:
C
C         DO UNTIL(ALL VECTORS GIVEN "HIGH"
C                  MEMBERSHIP IN CLASS 2)
          DO 9 VECTOR=1,VCOUNT
           MFUNCT(1,VECTOR)=LOW
           MFUNCT(2,VECTOR)=HIGH
```

```
            MFUNCT(3,VECTOR)=LOW
9           CONTINUE
C           END DO UNTIL
C
          IF(KLASES.LE.2) THEN
C
C       GIVE VECTOR NUMBER "FAR1" A "HIGH"
C        MEMBERSHIP IN CLASS 1
C
             MFUNCT(1,FAR1)=HIGH
             MFUNCT(2,FAR1)=LOW
             MFUNCT(3,FAR1)=LOW
C
          ELSE IF(KLASES.GE.3) THEN
C
C       GIVE VECTOR NUMBER "FAR1" HIGH
C        MEMBERSHIP IN CLASS 3
C
             MFUNCT(1,FAR1)=LOW
             MFUNCT(2,FAR1)=LOW
             MFUNCT(3,FAR1)=HIGH
C
C       GIVE VECTOR NUMBER "FAR2" HIGH
C        MEMBERSHIP IN CLASS 1
C
             MFUNCT(1,FAR2)=HIGH
             MFUNCT(2,FAR2)=LOW
             MFUNCT(3,FAR2)=LOW
C
          END IF
C
          RETURN
          END
```

```
$OPTIMIZE
C   A SUBROUTINE WHICH OUTPUTS THE MEMBERSHIP FUNCTION
C   ARRAY SELECTED VIA VALUE OF "CHOICE" AND "OPTION".
C
C   WRITTEN BY: MICHAEL R. GRAY
C
C   COMPLETED: MAY 84
C
C   FILENAME: MGMEMBPR.FTN
C
C   CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C    CALL MEMBPR(KLASES,VCOUNT,LU,CHOICE,OPTION)
C
C   INPUT VARIABLES(NOT CHANGED):
C
C    KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C    VCOUNT - INTEGER COUNT OF SAMPLES IN DATA SET
C
C    LU - INTEGER WHICH HOLDS THE LOGICAL UNIT
C         NUMBER USED FOR OUTPUT
C
C    CHOICE - INTEGER WHICH SELECTS WHICH MEMBERSHIP
C             FUNCTION ARRAY TO OUTPUT
C
C    OPTION - INTEGER WHICH SELECTS WHETHER TO OUTPUT
C             ENTIRE MEMBERSHIP ARRAY "NMFUNC" OR ONLY THE
C             PORTION WHICHINCLUDES MISCLASSIFIED SAMPLES
C
C    MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS MEMBERSHIPS
C             PRODUCED EITHER BY A CLUSTERING ALGORITHM
C             OR DIRECT ASSIGNMENT, PASSED IN LABELLED
C             FORTRAN COMMON "AREA3"
C
C    NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS MEMBERSHIP
C             ASSIGNMENTS PRODUCED BY ONE OF THE
C             CLASSIFER ALGORITHMS, PASSED IN
C             LABELLED FORTRAN COMMON "AREA5"
C
C    WRONG - INTEGER*2 ARRAY OF DIMENSION 242 WHICH
C            HOLDS THE INDICES OF MISCLASSIFIED VECTORS,
C            PASSES IN LABELLED FORTRAN COMMON "AREA4"
C
C    WRNCNT - INTEGER COUNT OF MISCLASSIFIED SAMPLES
C             WHOSE INDICES ARE STORED IN "WRONG",
C             PASSED IN LABELLED FORTRAN COMMON "AREA4"
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          PSEUDO - CODE
C
C   ENTER MEMBPR
C     IF(CHOICE <= 2 ) THEN
C       OUTPUT HEADING FOR MEMBERSHIP ARRAY
C        PRODUCED BY CLUSTERING ALGORITHM
C     ELSE
C       OUTPUT HEADING FOR MEMBERSHIP ARRAY
C        PRODUCED BY CLASSIFICATION ALGORITHM
C     END IF
C     IF(OPTION = 1 ) THEN
C       IF(CHOICE <= 2) THEN
C         OUTPUT MEMBERSHIP ARRAY PRODUCED
C          BY CLUSTERING ALGORITHM
C       ELSE
C         OUTPUT MEMBERSHIP ARRAY PRODUCED
C          BY CLASSIFICATION ALGORITHM
C       END IF
C     ELSE
C       IF(CHOICE <= 2) THEN
C         OUTPUT MEMBERSHIPS OF SAMPLES MISCLASSIFIED
C          BY CLUSTERING ALGORITHM
C       ELSE
C         OUTPUT MEMBERSHIPS OF SAMPLES MISCLASSIFIED
C          BY CLASSIFICATION ALGORITHM
```

```
C       END IF
C     END IF
C   RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE MEMBPR(KLASES,VCOUNT,LU,CHOICE,OPTION)
C
      REAL MFUNCT(3,242),NMFUNC(3,242)
      INTEGER VCOUNT,CLASS,CHOICE,OPTION,WRNCNT,COUNT
      INTEGER*2 WRONG(242)
      COMMON /AREA3/MFUNCT /AREA4/WRNCNT,WRONG
      COMMON /AREA5/NMFUNC
C
      IF(CHOICE.LE.2) THEN
        WRITE(LU,1)
1       FORMAT(/,43X,'PROTOTYPE DATA MEMBERSHIP'
     1         ' FUNCTION ARRAY')
      ELSE
        WRITE(LU,2)
2       FORMAT(/,43X,'NEWLY ASSIGNED MEMBERSHIP'
     1         ' FUNCTION ARRAY')
      END IF
      IF(OPTION.EQ.1) THEN
        WRITE(LU,3)
3       FORMAT(' ','CLASS')
        DO 11 INDEX=1,VCOUNT,20
          IF((INDEX+20).GT.VCOUNT) THEN
            COUNT=VCOUNT
          ELSE
            COUNT=INDEX+19
          END IF
          WRITE(LU,4) (I,I=INDEX,COUNT)
4         FORMAT(' ',6X,'X',1X,20(2X,I3,1X))
          DO 8 CLASS=1,KLASES
            IF(CHOICE.LE.2) THEN
              WRITE(LU,7) CLASS,(MFUNCT(CLASS,I),
     1                     I=INDEX,COUNT)
7             FORMAT(' ',I3,4X,20F6.3)
            ELSE
              WRITE(LU,7) CLASS,(NMFUNC(CLASS,I),
     1                     I=INDEX,COUNT)
            END IF
8         CONTINUE
11      CONTINUE
C
      ELSE
C
        WRITE(LU,12)
12      FORMAT(/,50X,'MISCLASSIFIED VECTORS ONLY')
        WRITE(LU,3)
C
        DO 18 INDEX=1,WRNCNT,20
          IF((INDEX+20).GT.WRNCNT) THEN
            COUNT=WRNCNT
          ELSE
            COUNT=INDEX+19
          END IF
          WRITE(LU,4) (WRONG(I),I=INDEX,COUNT)
          DO 14 CLASS=1,KLASES
            IF(CHOICE.LE.2) THEN
              WRITE(LU,7) CLASS,(MFUNCT(CLASS,
     1              WRONG(I)),I=INDEX,COUNT)
            ELSE
              WRITE(LU,7) CLASS,(NMFUNC(CLASS,
     1              WRONG(I)),I=INDEX,COUNT)
            END IF
14        CONTINUE
18      CONTINUE
      END IF
C
      RETURN
      END
```

```
$OPTIMIZE
C  A SUBROUTINE WHICH COMPUTES THE UPPER AND
C  LOWER CUT-SETS OF A FUZZY MEMBERSHIP
C  FUNCTION ARRAY AND OUTPUTS THE COUNTS OF
C  SAMPLE VECTORS IN THE RESULTING CUT-SETS.
C
C  WRITTEN BY: MICHAEL R. GRAY
C
C  COMPLETED: APR 84
C
C  FILENAME: MGCUTSET.FTN
C
C  CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C   CALL CUTSET(ALPHA,BETA,KLASES,VCOUNT,CHOICE,LU)
C
C  INPUT VARIABLES(NOT CHANGED):
C
C   ALPHA - REAL VALUE OF THE UPPER MEMBERSHIP
C           LIMIT FOR THE CUT-SET TO BE ASIGNED
C
C   BETA - REAL VALUE OF THE LOWER MEMBERSHIP
C          LIMIT FOR THE CUT-SET TO BE ASIGNED
C
C   KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C   VCOUNT - INTEGER COUNT OF NUMBER OF SAMPLE VECTORS
C
C   CHOICE - INTEGER WHICH CHOOSES BETWEEN CUT-SETS
C            OF MEMBERSHIPS IN ARRAY ASSIGNED BY
C            CLUSTERING OR CLASSIFICATION ALGORITHM
C
C   LU - INTEGER VALUE WHICH SETS THE LOGICAL
C        UNIT FOR OUTPUT
C
C   MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C            MEMBERSHIPS COMPUTED BY A CLUSTERING
C            ALGORITHM, PASSED IN LABELLED
C            FORTRAN COMMON "AREA3"
C
C   NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS THE
C            MEMBERSHIPS COMPUTED BY A CLASSIFICATION
C            A CLASSIFICATION ALGORITHM, PASSED IN
C            LABELLED FORTRAN COMMON "AREA5"
C
C   WRONG - INTEGER*2 ARRAY OF DIMENSION 242 WHICH HOLDS
C           THE INDICES OF MISCLASSIFIED SAMPLES, PASSED
C           IN LABELLED FORTRAN COMMON "AREA4"
C
C   WRNCNT - INTEGER WHICH SPECIFIES NUMBER OF
C            MISCLASSIFIED SAMPLES INDICES IN "WRONG",
C            PASSED IN LABELLED FORTRAN COMMON "AREA4"
C
C  OUTPUT VARIABLES:
C
C   KALPHA - INTEGER*2 ARRAY 3 BY 242 WHICH HOLDS
C            INIDCES OF SAMPLES WITH MEMBERSHIP
C            > "ALPHA" FOR EACH CLASS
C
C   ACOUNT - INTEGER ARRAY OF DIMENSION 3 WHICH HOLDS
C            COUNT OF SAMPLES IN "KALPHA" FOR EACH CLASS
C
C   KBETA - INTEGER*2 ARRAY 3 BY 242 WHICH HOLDS
C           INDICES OF SAMPLES WITH MEMBERSHIP
C           < "BETA" FOR EACH CLASS
C
C   BCOUNT - INTEGER ARRAY OF DIMENSION 3 WHICH HOLDS
C            COUNT OF SAMPLES IN "KBETA" FOR EACH CLASS
C
C   BETWEN - INTEGER*2 ARRAY 3 BY 242 WHICH HOLDS INDICES
C            OF SAMPLES WITH MEMBERSHIP >= "BETA" AND
C            <= "ALPHA" FOR EACH CLASS
C
C   ICOUNT - INTEGER ARRAY OF DIMENSION 3 WHICH HOLDS
C            COUNT OF SAMPLES IN "BETWEN" FOR EACH CLASS
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                        PSEUDO - CODE SOLUTION
C
C  ENTER CUTSET
C    PROMPT USER TO CHOOSE IF ALL SAMPLES TO BE
C     TESTED ("ICHOICE"=1) OR ONLY MISCLASSIFIED
C     SAMPLES TO BE TESTED("ICHOICE"=2)
C    IF(ICHOICE=2) THEN
C      SET SAMPLE COUNT TO "WRNCNT"
C    ELSE
C      SET SAMPLE COUNT TO "VCOUNT"
C    END IF
C    DO UNTIL( ALL CLASSES CONSIDERED)
C      IF( CHOICE <=2 ) THEN
C        GET INDICES AND COUNT OF SAMPLES WITH
C         MEMBERSHIPS > "ALPHA" IN CURRENT
C         CLASS USING "MFUNCT" MEMBERSHIPS
C        GET INDICES AND COUNT OF SAMPLES WITH
C         MEMBERSHIPS < "BETA" IN CURRENT
C         CLASS USING "MFUNCT" MEMBERSHIPS
C        GET INDICES AND COUNT OF SAMPLES WITH
C         MEMBERSHIP <= "ALPHA" AND >= "BETA IN
C         CURRENT CLASS USING "MFUNCT" MEMBERSHIPS
C      ELSE
C        GET INDICES AND COUNT OF SAMPLES WITH
C         MEMBERSHIPS > "ALPHA" IN CURRENT
C         CLASS USING "NMFUNC" MEMBERSHIPS
C        GET INDICES AND COUNT OF SAMPLES WITH
C         MEMBERSHIPS < "BETA" IN CURRENT
C         CLASS USING "NMFUNC" MEMBERSHIPS
C        GET INDICES AND COUNT OF SAMPLES WITH
C         MEMBERSHIPS <= "ALPHA" AND >= "BETA" IN
C         CURRENT CLASS USING "NMFUNC" MEMBERSHIPS
C      END IF
C    END DO UNTIL
C    OUTPUT TO "LU" THE COUNTS OF SAMPLES IN THE
C     UPPER, LOWER, AND INNER CUT-SETS
C  RETURN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE CUTSET(ALPHA,BETA,KLASES,VCOUNT,CHOICE,LU)
C
      REAL MFUNCT(3,242),NMFUNC(3,242)
      INTEGER VCOUNT,CLASS,VECTOR,CHOICE
      INTEGER ACOUNT(3),BCOUNT(3),ICOUNT(3),WRNCNT
      INTEGER*2 KALPHA(3,242),KBETA(3,242),BETWEN(3,242)
      INTEGER*2 WRONG(242)
      COMMON /AREA3/MFUNCT /AREA4/WRNCNT,WRONG /AREA5/NMFUNC
      COMMON /AREA8/ACOUNT,BCOUNT,ICOUNT,KALPHA,KBETA,BETWEN
C
       WRITE(5,1)
1      FORMAT(/,2X,'ENTER YOUR CHOICE:',//,5X,
     1'1 - CONSIDER ALL VECTORS',/,5X,
     2'2 - CONSIDER ONLY MISCLASSIFIED VECTORS')
       READ(5,2) ICHOICE
2      FORMAT(I1)
C
       IF(ICHOICE.EQ.2) THEN
         NUMBER=WRNCNT
       ELSE
         NUMBER=VCOUNT
       END IF
C
       DO 12 CLASS=1,KLASES
        ACOUNT(CLASS)=0
        BCOUNT(CLASS)=0
        ICOUNT(CLASS)=0
C
        IF(CHOICE.LE.2) THEN
C         DO UNTIL(CHOSEN VECTORS OF DATA SET
C                  FOR CURRENT CLASS CHECKED)
```

```
            DO 4 INDEX=1,NUMBER
              IF(ICHOICE.EQ.2) THEN
                VECTOR=WRONG(INDEX)
              ELSE
                VECTOR=INDEX
              END IF
              IF(MFUNCT(CLASS,VECTOR).GT.ALPHA) THEN
                ACOUNT(CLASS)=ACOUNT(CLASS)+1
                KALPHA(CLASS,ACOUNT(CLASS))=VECTOR
              ELSE IF(MFUNCT(CLASS,VECTOR).LT.BETA) THEN
                BCOUNT(CLASS)=BCOUNT(CLASS)+1
                KBETA(CLASS,BCOUNT(CLASS))=VECTOR
              ELSE
                ICOUNT(CLASS)=ICOUNT(CLASS)+1
                BETWEN(CLASS,ICOUNT(CLASS))=VECTOR
              END IF
4           CONTINUE
C           END DO UNTIL
          ELSE
C           DO UNTIL(ALL VECTORS OF TEST SET
C                    FOR CURRENT CLASS CHECKED)
            DO 6 INDEX=1,NUMBER
              IF(ICHOICE.EQ.2) THEN
                VECTOR=WRONG(INDEX)
              ELSE
                VECTOR=INDEX
              END IF
              IF(NMFUNC(CLASS,VECTOR).GT.ALPHA) THEN
                ACOUNT(CLASS)=ACOUNT(CLASS)+1
                KALPHA(CLASS,ACOUNT(CLASS))=VECTOR
              ELSE IF(NMFUNC(CLASS,VECTOR).LT.BETA) THEN
                BCOUNT(CLASS)=BCOUNT(CLASS)+1
                KBETA(CLASS,BCOUNT(CLASS))=VECTOR
              ELSE
                ICOUNT(CLASS)=ICOUNT(CLASS)+1
                BETWEN(CLASS,ICOUNT(CLASS))=VECTOR
              END IF
6           CONTINUE
C           END DO UNTIL
          END IF
C
12      CONTINUE
C       END DO UNTIL
C
        IF(ICHOICE.EQ.2) THEN
          WRITE(LU,13)
13        FORMAT(/,' THE FOLLOWING CONSIDERS '
     1             'MISCLASSIFIED VECTORS.')
        ELSE
          WRITE(LU,14)
14        FORMAT(/,' THE FOLLOWING CONSIDERS '
     1             'ALL VECTORS IN DATA SET.')
        END IF
C
        DO 16 ICLASS=1,KLASES
          WRITE(LU,15) ACOUNT(ICLASS),ALPHA,ICLASS
15        FORMAT(/,4X,I3,' VECTORS WITH MEMBERSHIP > ',
     1F5.2,' IN CLASS',I2)
16      CONTINUE
        DO 18 ICLASS=1,KLASES
          WRITE(LU,17) BCOUNT(ICLASS),BETA,ICLASS
17        FORMAT(/,4X,I3,' VECTORS WITH MEMBERSHIP < ',
     1F5.2,' IN CLASS',I2)
18      CONTINUE
        DO 20 ICLASS=1,KLASES
          WRITE(LU,19) ICOUNT(ICLASS),ALPHA,BETA,ICLASS
19        FORMAT(/,4X,I3,' VECTORS WITH MEMBERSHIP >= ',
     1F5.2,' AND <= ',F5.2,' IN CLASS',I2)
20      CONTINUE
C
        RETURN
        END
```

```
$OPTIMIZE
C  A SUBROUTINE WHICH COMPUTES AND OUTPUTS
C  A CONFUSION MATRIX OF THE HARD PARTITION
C  WHICH RESULTS FROM ASSIGNING A DATA
C  VECTOR TO A CLASS IN WHICH IT HAS
C  MEMBERSHIP FUNCTION VALUE.
C
C  WRITTEN BY: MICHAEL R. GRAY
C
C  COMPLETED:APR 84
C
C  FILENAME: MGCMTRIX.FTN
C
C  CALLING SEQUENCE(FROM A FORTRAN ROUTINE):
C   CALL CMTRIX(KLASES,VCOUNT,LU,CHOICE)
C
C  INPUT VARIABLES(NOT CHANGED):
C
C   KLASES - INTEGER COUNT OF NUMBER OF CLASSES
C
C   VCOUNT - INTEGER COUNT OF NUMBER OF DATA SAMPLES
C
C   LU - INTEGER WHICH HOLDS LOGICAL UNIT
C        NUMBER FOR OUTPUT
C
C   CHOICE - INTEGER WHICH SELECTS WHETHER TO CONSIDER
C            MEMBERSHIPS PRODUCED BY CLUSTERING
C            ALGORITHM(=2) OR CLASSIFICATION ALGORITHM(=1)
C
C   MFUNCT - REAL ARRAY 3 BY 242 WHICH HOLDS MEMBERSHIPS
C            ASSIGNED BY CLUSTERING ALGORITHM, PASSED
C            IN LABELLED FORTRAN COMMON "AREA3"
C
C   NMFUNC - REAL ARRAY 3 BY 242 WHICH HOLDS MEMBERSHIPS
C            ASSIGNED BY CLASSIFICATION ALGORITHM, PASSED
C            IN LABELLED COMMON "AREA5"
C
C  OUTPUT VARIABLES:
C
C   WRONG - INTEGER*2 ARRAY OF DIMENSION WHICH HOLDS
C           INDICES OF MISCLASSIFIED SAMPLES, PASSED
C           IN LABELLED FORTRAN COMMON "AREA4"
C
C   WRNCNT - INTEGER WHICH HOLDS THE COUNT OF
C            MISCLASSIFIED SAMPLES, PASSED IN
C            IN LABELLED FORTRAN COMMON "AREA4"
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                        PSEUDO - CODE
C
C  ENTER CMTRIX
C   INITIALIZE "WRNCNT" TO 0
C   INITIALIZE VECTOR INDEX TO 1
C   DO UNTIL(CLASSIFICATION OF ALL SAMPLES DETERMINED
C     DETERMINE ACTUAL CLASS OF CURRENT VECTOR
C     IF(CHOICE <= 2) THEN
C       DETERMINE CLASS OF MAXIMUM MEMBERSHIP
C        FOR CURRENT VECTOR FROM "MFUNCT" ARRAY
C     ELSE
C       DETERMINE CLASS OF MAXIMUM MEMBERSHIP
C        FOR CURRENT VECTOR FROM "NMFUNC" ARRAY
C     END IF
C     INCREMENT CONFUSION MATRIX ELEMENT (ACTUAL,MAXIMUM)
C     IF(ACTUAL NOT EQUAL MAXIMUM) THEN
C       INCREMENT "WRNCNT"
C       PUT INDEX OF MISCLASSIFIED SAMPLE IN "WRONG" ARRAY
C     END IF
C     INCREMENT VECTOR INDEX
C   END DO UNTIL
C   OUTPUT RESULTANT CONFUSION MATRIX
C RETURN
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE CMTRIX(KLASES,VCOUNT,LU,CHOICE)
C
      REAL MFUNCT(3,242),NMFUNC(3,242)
      INTEGER CARRAY(3,3),VCOUNT,VECTOR,CLASS
      INTEGER ACTUAL,CHOICE,WRNCNT,START(3),END(3)
      INTEGER*2 WRONG(242)
      COMMON /AREA3/MFUNCT /AREA5/NMFUNC
      COMMON /AREA6/START,END /AREA4/WRNCNT,WRONG
C
C     DO UNTIL(CONFUSION MATRIX "ZEROED")
      DO 2 I=1,KLASES
        DO 1 J=1,KLASES
          CARRAY(I,J)=0
1       CONTINUE
2     CONTINUE
C     END DO UNTIL
C
      ERRMAX=0.0
      WRNCNT=0
C
C     DO UNTIL(ALL VECTORS ASSIGNED TO A CLASS)
      DO 4 VECTOR=1,VCOUNT
C
        DMAX=0.0
C
        DO 3 CLASS=1,KLASES
C
C  FIND ACTUAL CLASS VECTOR BELONGS IN
C
          IF((VECTOR.GE.START(CLASS)).AND.
     1              (VECTOR.LE.END(CLASS))) THEN
            ACTUAL=CLASS
          END IF

C  FIND CLASS OF MAXIMUM MEMBERSHIP
C
          IF(CHOICE.LE.2) THEN
            IF(MFUNCT(CLASS,VECTOR).GT.DMAX) THEN
              DMAX=MFUNCT(CLASS,VECTOR)
              NUMBER=CLASS
            END IF
C
          ELSE
            IF(NMFUNC(CLASS,VECTOR).GT.DMAX) THEN
              DMAX=NMFUNC(CLASS,VECTOR)
              NUMBER=CLASS
            END IF
C
          END IF
C
3       CONTINUE
C
        CARRAY(ACTUAL,NUMBER)=CARRAY(ACTUAL,NUMBER)+1
C
        IF(NUMBER.NE.ACTUAL) THEN
          WRNCNT=WRNCNT+1
          WRONG(WRNCNT)=VECTOR
        END IF
C
4     CONTINUE
C     END DO UNTIL
C
      WRITE(LU,5)
5     FORMAT(/,' THE HARD PARTITION SHOWN IN THE '
     1'CONFUSION MATRIX WAS CONSTRUCTED USING MAXIMUM '
     2'MEMBERSHIP VALUE FOR EACH CLASS.')
C
      WRITE(LU,6) KLASES,KLASES
6     FORMAT(' ',7X,'CONFUSION MATRIX: ROWS 1-',I2,
     1' SHOW CLASSIFICATION OF CLASSES 1-',I2,
     2',',' RESPECTIVELY.')
```

```
      WRITE(LU,7) (I,I=1,KLASES)
7     FORMAT(/,11X,'CLASS',3I5)
C
C     DO UNTIL(CONFUSION MATRIX PRINTED)
      DO 9 INDEX=1,KLASES
       WRITE(LU,8) INDEX,(CARRAY(INDEX,I),I=1,KLASES)
8      FORMAT(/,13X,I1,4X,I3,2X,I3,2X,I3)
9     CONTINUE
C     END DO UNTIL
C
      RETURN
      END
```

VITA

Michael R. Gray was born on September 10, 1954 in Saint Louis, Missouri. He received the B.S. degree in electrical engineering and the B.S. degree in computer engineering from the University of Missouri-Columbia in 1983 and the M.S. degree in electrical engineering from the University of Missouri-Columbia in 1984.

END

FILMED

10-84

DTIC